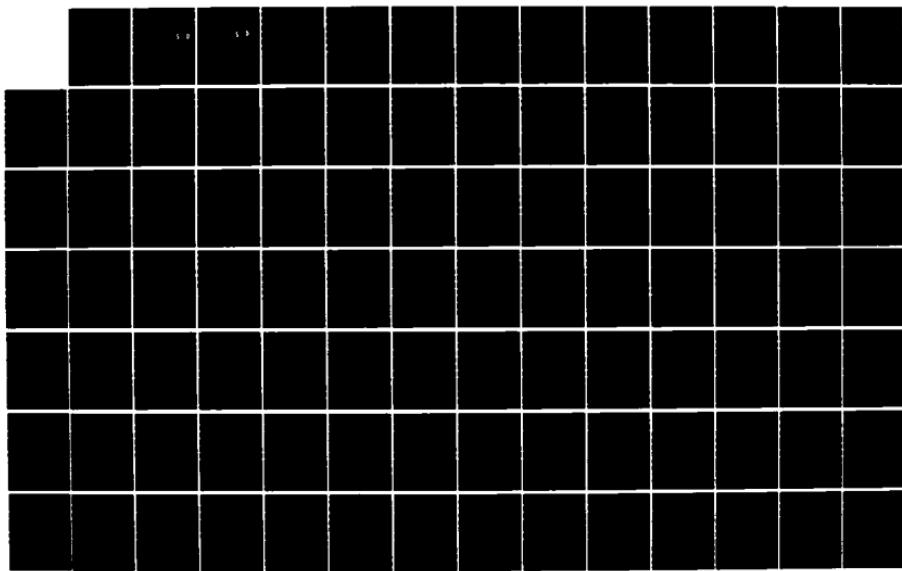


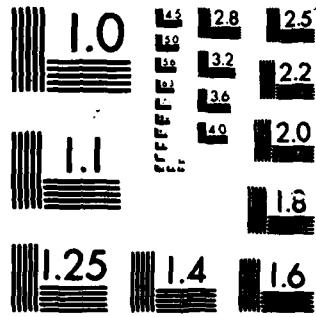
AD-A163 966 DEVELOPMENT OF A DIGITAL INTERACTIVE CONTROLLER
EVALUATION SYSTEM (DICES)(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
UNCLASSIFIED S B ECKERT DEC 85 AFIT/GE/ENG/85D-13

1/3

F/G 9/5

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

AD-A163 966

DTIC FILE COPY



1
DTIC
SELECTED
FEB 12 1986
S D

DEVELOPMENT OF A DIGITAL INTERACTIVE
CONTROLLER EVALUATION SYSTEM (DICES)

THESIS

Scott B. Eckert
Captain, USAF

AFIT/GE/ENG/85D-13

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

36 2 11 - 13

AFIT/GE/ENG/85D-13

1

DTIC
SELECTED
FEB 12 1986
S D

DEVELOPMENT OF A DIGITAL INTERACTIVE
CONTROLLER EVALUATION SYSTEM (DICES)

THESIS

Scott B. Eckert
Captain, USAF

AFIT/GE/ENG/85D-13

Approved for public release; distribution unlimited

AFIT/GE/ENG/85D-13

DEVELOPMENT OF A
DIGITAL INTERACTIVE CONTROLLER
EVALUATION SYSTEM (DICES)

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Scott B. Eckert, B.S.
Captain, USAF

December 1985

Accession For	
NTIS	CRA&I
DTIC	TAB
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

Approved for public release; distribution unlimited



PREFACE

This investigation details the development of a Digital Interactive Controller Evaluation System (DICES). DICES permits the interactive implementation of a digital controller that has been designed using a control systems CAD package. Performance testing of the controller is then performed in a closed-loop system using programmable test instrumentation under control of a VAX 11/780.

I would like to thank Mr. Robert Durham, EE laboratory supervisor, and all of the superb staff for the assistance provided during the period of this investigation. Mr. Robert Ewing provided invaluable assistance in using the AFIT Information Sciences Laboratory's VAX 11/780.

Thanks also goes to Texas Instruments, Inc. for the many helpful documents provided and to Mr. Rick Spearnock of Brueel and Kjaer Instruments.

I wish to express my gratitude to Dr. Gary Lamont, my thesis advisor, for his guidance and encouragement throughout this investigation. I also want to thank the other members of my thesis committee, Capt. Richard Linderman, Mr. Richard McKinney, and especially Capt. Dave King, for his valuable comments to early draft versions of this thesis. Finally, I would like to express my appreciation to my two friends in Los Angeles, CA for their support throughout the course of my graduate study.

Scott B. Eckert

Contents

Preface	iii
Table of Contents	iv
List of Figures	xiii
List of Tables	xvi
List of Symbols and Definitions	xvii
Abstract	xix
I. Introduction	1
Background	3
Statement of Problem	3
Approach	4
System Concept	5
Literature Search	9
Scope of Thesis Effort	9
Structure	10
II. Requirements Analysis and Definition	10
Introduction	10
System Design Requirements	10
User Interface	12
Program Control	13
Data Base Control	13
Error Protection	14
User Assistance - HELP	14
Modularity	15
Documentation	16

Functional Analysis and Requirements Definition	17
Top-Level Functional Flow Description	18
Detailed Functional Analysis	23
Level 1.0	23
Level 2.0	23
Level 3.0	23
Level 4.0	26
Level 5.0	30
Level 6.0	37
Functional Requirements Allocation	40
III. System Design	47
Introduction	47
System Design Constraints	47
EVM/AIB Board	47
Computer System	48
Trade-off Summary	50
System Component Description	50
TMS32010 EVM/AIB	50
System Development Tools	53
VAX 11/780	55
B/K 2032 System Analyzer	55
EAI TR-48 Analog Computer	56
Design Overview	56

Detailed Module Design	61
Detailed User Interface Design	61
Level 1.0 System Performance Specification	61
Level 2.0 Develop Plant Model	62
Level 3.1 Display User Options	62
Level 3.3 Test Plant Simulation	62
Level 3.4 Connect Controller	63
Level 4.2 Design Techniques	63
Level 4.3 Analyze Performance	63
Level 4.4 Store Coefficients	63
Level 5.1 Retrieve Coefficients	64
MEMORY.DAT Organization	65
Level 5.5 Re-analyze	69
Level 6.1 Test Requirements	69
Level 6.5 Collect Test Results	71
Detailed Filter Implementation Design	72
Level 5.2 Filter Structure	72
Default Structure	72
Selection of Default Structure	73
Maximum Order	76
Level 5.3 Pole/Zero Pairing	76
Level 5.4 Quantize Coefficients	82
Quantization	83
Level 5.6 Scaling and Ordering	92

Level 5.7 Sample Period Selection	92
Hardware Limitations	93
Level 5.8 Generate Code	99
Assembly	100
Detailed Plant Modelling Design	101
Level 3.2 Patch-up Equations	101
Level 3.4 Connect System	103
Evaluation Module	105
IEEE-488 Interfaces	105
Signal Generator	105
B/K 2032 System Analyzer	106
Cascade Controller	106
TMS32010	106
Plant	107
Feedback Controller	107
TMS32010	107
Plant	107
Detailed System Configuration Design	108
Level 4.1 Access CAD Package	108
Level 5.9 Load TMS3210 Object Code	108
Download Process	108
Level 6.1 Determine Test Reqs	109
Default Instrument Settings	109
B/K Settings	109

Wavetek 172B Settings	111
Detailed Performance Evaluation Design	112
Level 6.2 Generate Command Strings	112
General Programming Description	112
Wavetek 172B	113
B/K 2032	114
Step Response	116
Frequency Response	119
Level 6.3 Send Command String	120
IEEE-488 Bus Description	122
VAX 11/780-IEEE-488 Interface	123
IEEE-488 Bus Extender	125
High-Level Software Interfaces	126
Error Codes	129
Level 6.4 Start Test Sequence	129
Level 6.5 Collect Test Results	131
Hardware System Design	132
User Interface	133
Analog Computer Interface	134
Performance Analysis	135
IV. Implementation	139
Introduction	139

System Implementation	139
User Interface	140
Main Menu	140
Level 1.0 Performance Specs	142
Level 2.0 Develop Plant Model	143
Level 3.1 Display Assistance	143
Level 3.3 Test Simulation	144
Level 3.4 Connect System	144
Level 4.2 Design Controller	144
Level 4.3 Analyze Performance	145
Level 4.4 Store Coefficients	145
Level 5.1 Retrieve Factors	145
Level 5.5 Re-analyze Performance	148
Level 6.1 Test Requirements	148
Level 6.5 Collect Test Results	149
Step Response	149
Frequency Response	149
Filter Implementation	150
Level 5.2 Select Filter Structure	151
3DFILT.THS Implementation	151
Level 5.3 Pole/Zero Pairing	155
Subroutines	155
ANYCOMPZEROS	155
ANYREALZEROS	159
MARK1Z	160

	MARK2Z	161
	CLOSESTZERO	161
	CLOSESTREALPAIRC	162
	CLOSESTCOMPLEXPAIR	162
	CLOSESTREALPAIRR	163
	Level 5.4 Quantize Coefficients	163
	Level 5.6 Scaling and Ordering	165
	Level 5.7 Sample Period Selection	166
	Level 5.8 Generate TMS32010 Code	166
	Plant Modelling	167
	Level 3.2 Patch-up Equations	167
	Level 3.4 Connect System	167
	System Configuration	168
	Level 4.1 Access CAD Package	169
	Level 5.9 Load TMS32010 Object Code ..	169
	Performance Evaluation	170
	Level 6.2 Generate Command String	172
	Step Response	172
	Frequency Response	175
	Level 6.3 Send Command String	176
	Level 6.4 Start Test Sequence	177
	Summary	178
V.	Test	179
	Introduction	179
	Discussion of Test Strategies	179

Top-Down Testing	180
Bottom-Up Testing	180
General test Plan	181
Software Test Plan	182
Hardware Test Plan	183
System Test Plan	183
Problem Statement	183
Test Results	185
Software Test Results	185
Hardware Test Results	185
Filter Modifications	187
Sample and Hold	187
System Test Results	189
Test Procedure and Results	189
System Connection	189
VAX Sign-on	190
Step 1 - System Performance Requirements	190
Step 2 - Develop Plant Model	191
Step 3 - Simulate Plant	193
Step 4 - Controller Design	199
Step 5 - Filter Implementation ..	201
Step 6 - Performance Evaluation ..	204
Summary	210
VI. Conclusions and Recommendations	211
Conclusions	211

Recommendations for Further Investigation	212
Bibliography	215
Appendix A: User's Manual	219
Appendix B: Data Dictionary	296
Appendix C: Data Flow Diagrams	322
Appendix D: Test Results	327
Appendix E: Program Listings	360
Appendix F: Schematic Diagrams	415
Vita	417

List of Figures

<u>Figure</u>		<u>Page</u>
1	Hybrid Control Simulation	2
2	System Concept	6
3	DICES Functional Diagram	8
4	CAD Perspective	12
5	System Design Requirements	13
6	Functional Flow Diagram	20
7	Level 3.0 - Plant Simulation	25
8	Level 4.0 - Design Controller	28
9	Design and Analysis Methods	30
10	Level 5.0 - Implement Controller	32
11	Level 6.0 - Closed-Loop Testing	39
12	TMS32010 Architecture	52
13	3D Filter Structure	74
14	Example Filter	78
15	Pairing Algorithm	80
16	Pairing Algorithm	81
17	Circles of Radius $[a_2^q]^{1/2}$	85
18	Vertical Lines	86
19	Intersection of Figures 17 and 18	87
20	Sample-Period Errors	98
21	Analog Simulation Diagram	102
22	System Block Diagram	104
23	IEEE-488 Software Interfaces	121

24	Typical VAX 11/780 System	124
25	IEEE-488 Bus Extenders	126
26	EVM/VAX Interface	133
27	Unity Feedback Closed-Loop	134
28	Feedback Controller Closed-Loop	135
29	System Block Diagram	137
30	Main Menu	141
31	Filter Implementation Menu	141
32	Performance Testing Sub-Menu	142
33	Levels 1.0 and 2.0 Flowchart	142
34	Display Assistance Flowchart	143
35	RDORDER.THS Flowchart	146
36	RDHTF.THS Flowchart	147
37	Flowchart of 3DFILT.THS Implementation ...	154
38	PAIR.THS Flowchart	156
39	PAIR.THS Flowchart	157
40	Subroutine ANYCOMPZEROS	159
41	Subroutine ANYREALZEROS	160
42	Subroutine CLOSEST1ZERO	162
43	General Test Sequence Flowchart	171
44	Servo Control System	184
45	Servomotor System	192
46	Analog Computer Simulation	195
47	Uncompensated Actual Closed-Loop Simulation	196

48	Uncompensated Theoretical Closed-Loop Simulation	196
49	Pole/Zero Pairing	202
50	Second-Order Section Data	203
51	Compensated Theoretical Step Response ...	206
52	Actual Compensated Step Response	206
53	Theoretical Uncompensated Frequency Response Test (Basic System)	208
54	Test Results of Uncompensated System (Basic System)	208
55	Theoretical Compensated Frequency Response ($T_S = .05$)	209
56	Test Results of Compensated System ($T_S = .05$)	209

List of Tables

<u>Table</u>		<u>Page</u>
1	Level 3.0 Requirements Allocation	43
2	Level 4.0 Requirements Allocation	44
3	Level 5.0 Requirements Allocation	45
4	Level 6.0 Requirements Allocation	46
5	Trade-off Summary	49
6	UIM Requirements	58
7	FIM Requirements	59
8	PMM Requirements	59
9	SCM Requirements	60
10	PEM Requirements	61
11	MEMORY.DAT Organization	68
12	Array of Controller Roots	83
13	Sample-Period Errors	97
14	Source File Structure	99
15	Default Step-Response Settings	110
16	Default Wavetek Step-Response Settings ..	112
17	Device Numbers and Addresses	125
18	Step Response Theoretical Results	198
19	Step Response Test Results	198
20	Frequency Response Theoretical Results ..	210
21	Frequency Response Test Results	210

List of Symbols and Definitions

A/D	Analog-to-Digital
AIB	Analog Interface Board
ASCII	American Standard Code for Information Interchange
B/K	Brueel and Kjaer
CAD	Computer-Aided-Design
COB	Complementary-Offset-Binary
D/A	Digital-to-Analog
DC	Direct Current
DFD	Data Flow Diagrams
DICES	Digital Interactive Controller Evaluation System
DSP	Digital Signal Processing
EVM	Evaluation Module
f_{\max}	Maximum frequency accurately sampled
FORTRAN	FORmula TRANslator Programming Language
FFT	Fast Fourier Transform
$G_c(z)$	Cascade Controller Z-transfer function
$G_h(z)$	Feedback Controller Z-transfer function
$G_{hc}(z)$	Cascade or Feedback Controller Z-transfer function
$G_x(z)$	Uncompensated Plant Z-transfer function
ICECAP	Interactive Control Engineering Analysis Package (ICECAP)
IEEE-488	Institute of Electrical and Electronic Engineers Standard Digital Interface for Programmable Instruments

ISL	Information Sciences Laboratory
LSB	Least Significant Bit
LPM	Load Program Memory
LTA	Load T-Register and Accumulate
LTD	LTA + Shift Memory
MSB	Most Significant Bit
OB	Offset-Binary
SISO	Single-Input-Single-Output
SUT	System-Under-Test
t_p	Peak time
t_s	Settling time
T	Sample Period
TMS32010	Texas Instruments Microprocessor
TOTAL	Control System Design and Analysis CAD Package
VLSI	Very Large Scale Integration
ZOH	Zero-Order-Hold

Abstract

The Digital Interactive Controller Evaluation System (DICES) is a user interactive system which permits the implementation of digital controller designs based on the TMS32010 digital-signal-processing microprocessor for a given single input-single-output plant model. DICES partitions the controller design into second-order 3D filter sections and quantizes the coefficients. These coefficients are loaded into a generic filter program written in TMS32010 assembly language, which are then assembled and loaded into the TMS32010 for execution. The controller can be placed in the forward or feedback path of an analog computer system which reflects the plant model. Performance data is obtained via IEEE-488 controlled instruments under control of a VAX 11/780.

DEVELOPMENT OF A
DIGITAL INTERACTIVE CONTROLLER
EVALUATION SYSTEM (DICES)

I. Introduction

The use of digital filters/compensators offers many important engineering advantages, such as reproducibility and a guaranteed level of performance, increased ease in changing the filter characteristics, and the possibility of time-sharing the same hardware system among a multiplicity of filtering functions (14:ix). With the advent of high-speed, special-purpose microcomputers dedicated to digital processing applications, it is possible to implement complex, high sample-rate digital filters and digital signal processors (8). There are many computer-aided-design (CAD) packages available for analysis and design of such digital controllers and filters. However, there seldom exists a means to readily implement the design in order to test it on a realistic plant model. The performance will often vary substantially from the predicted theoretical performance because of effects of finite wordlength, arithmetic operations (e.g., truncation and rounding), and Analog-to-Digital (A/D) and Digital-to-Analog (D/A) converter errors. CAD design packages such as those referred to above typically assume a 'near infinite'

precision implementation which, of course, is not the 'real world'.

There are several hybrid computer simulation programs available (41) that simulate (via software) a continuous plant model and also provide simulation (again via software) of a finite word length digital controller placed in the closed-loop system. These are very useful, but do not allow for the 'real-world' interfacing of a real control algorithm executing on a real finite word length computer. There have been many hybrid simulations (via hardware) where a plant is simulated on an analog computer and the controller implemented on a digital computer as shown in figure 1 (11).

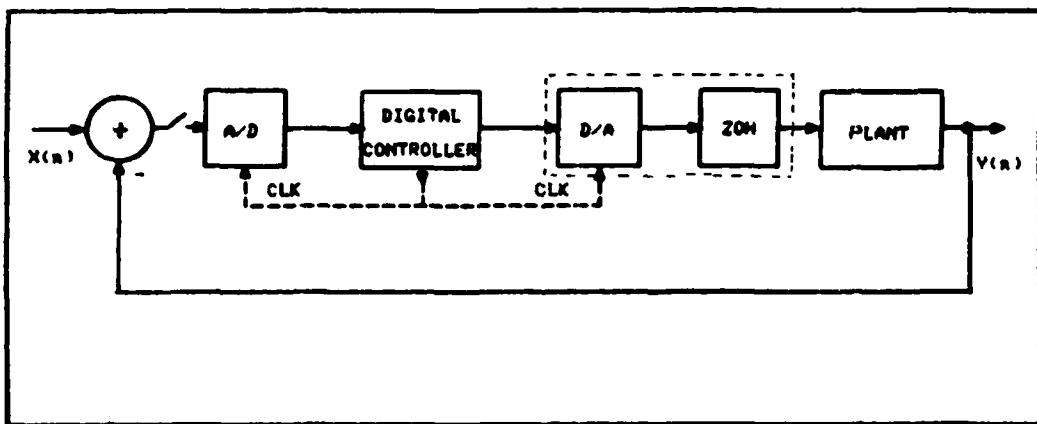


Figure 1. Hybrid Control Simulation

This is a very realistic 'simulation' because the only simulation is that of the continuous plant model on the analog computer, which can be of very 'high fidelity'. Thus, the hardware hybrid simulation is an excellent tool for studying the effects of implementing a digital controller on a finite word-length machine in order to control a plant that has been simulated on an analog computer.

Background

The AFIT Department of Electrical and Computer Engineering acquired a Texas Instruments TMS32010 Evaluation Module (EVM) and Analog Interface Board (AIB) for laboratory use. These two modules contain a TMS32010 microprocessor, Analog-to-Digital (A/D), and Digital-to-Analog (D/A) converter. There was also a wealth of digital signal processing (DSP) software available that executed on the AFIT Information Sciences Laboratory Vax 11/780. The laboratory also has a Bruei and Kjaer Model 2032 System Analyzer and a Wavetek 172B Signal Generator, both of which are capable of remote programming via an IEEE-488 interface. An LSI-11 was also available to function as a local controller. It was desired to integrate the EVM, AIB, B/K 2032 , and Wavetek 172B into an interactive digital controller evaluation system. This system could also incorporate an analog plant simulator and any applicable software packages available on the VAX 11/780.

Statement of Problem

The problem is to 1) develop requirements, 2) design, and, 3) implement a system which allows implementation of digital controller designs in a closed-loop Single-Input-Single-Output (SISO) system which has the plant characteristics modelled by an analog computer. The system must also incorporate a means to evaluate performance of the closed-loop system using standard figures-of-merit metrics (5:309-312). This system will be called the Digital Interactive Controller Evaluation System (DICES).

Approach

This thesis investigation consists of two major phases. The first consists of requirements definition and analysis for DICES. The second phase consists of the design and implementation of a subset of the requirements defined in phase one.

In Phase One, the overall system performance will be analyzed and requirements generated from this analysis. During the design phase, the requirements will be allocated to functional blocks within the system structure (Figure 3). This phase will generate requirements for a generic system of which portions will be implemented in Phase Two.

Phase Two's implementation subset will consist of a portion of each of the major requirements of Phase One.

The primary emphasis of this phase will be to implement DICES such that it provides a basic capability for digital controller implementation, as well as provide a basis for future expansion of the system.

System Concept

The Digital Interactive Controller Evaluation System (DICES) is a user interactive system which permits the implementation of digital controller designs on a state-of-the-art 16-bit microprocessor (Figure 2). This system also gives the user access to a computer-aided-design package (15,16) to design a digital controller based on a given continuous plant model. The user is free to use any of the techniques and/or approximations (6:354-421) that exist to design the digital controller. Once the design is complete, the user will be able to implement the controller algorithm on the high-speed, 16-bit TMS32010 microprocessor. The plant can be simulated using conventional analog computer circuits (35). A means to evaluate the controller's performance in a closed-loop mode will be provided using a remotely programmed system analyzer. The user can then compare the theoretical performance with the actual 'real-world' performance metrics obtained from DICES. This controller can then be inserted into a 'real-world' closed-loop control system, if desired.

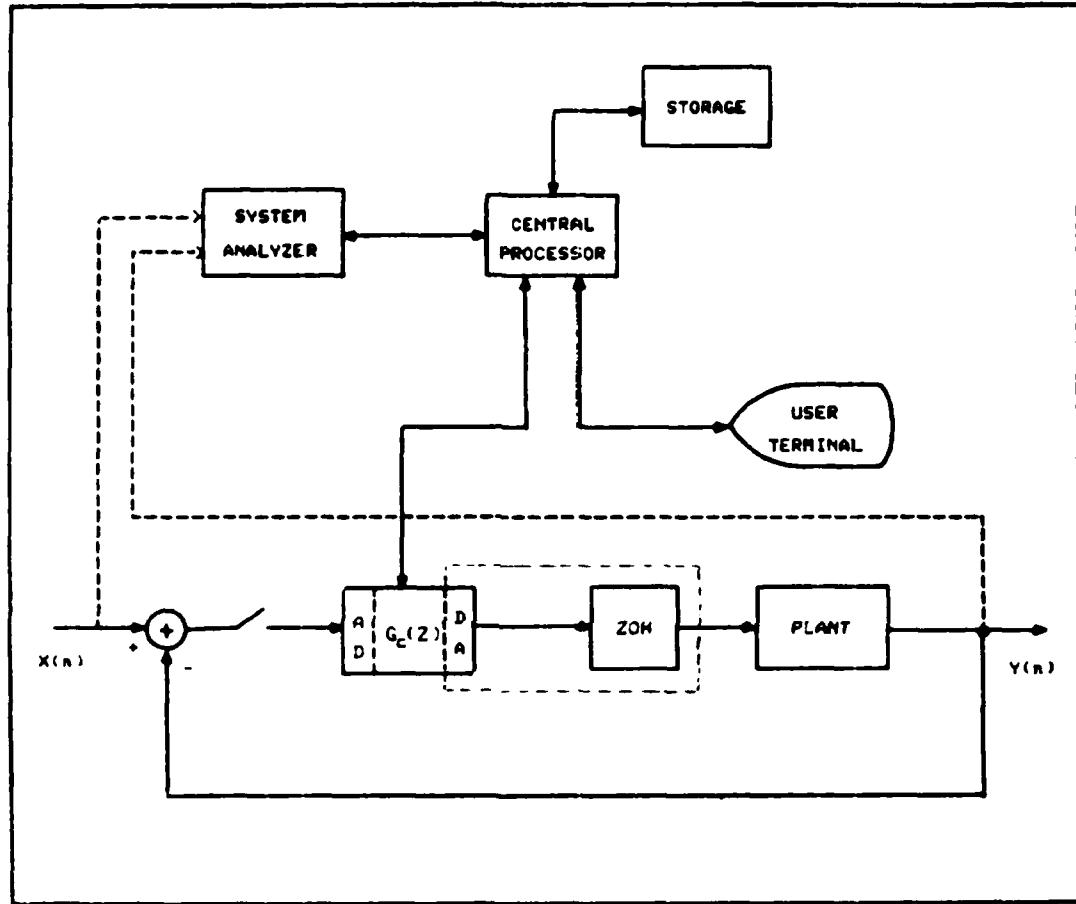


Figure 2. System Concept

DICES consists of five major functional areas (Figure 3).

a. User interface - Provides the user with menu-driven options, data storage, and results of tests.

b. System Configuration - Provides the control necessary to configure components of system for initialization, test, etc.

c. Filter Implementation - Provides the implementation of the controller design, $G_h(z)$ or $G_c(z)$ ($G_{hc}(z)$ means either), to be executed on the microprocessor. $G_h(z)$ is a feedback compensator while $G_c(z)$ is a cascade compensator.

d. Plant Modelling - Provides the user with a method of simulating a physical plant or process to be controlled.

e. Performance evaluation - Controls the tests on the System-Under-Test (SUT), which includes the controller (A/D, D/A, Zero-Order-Hold (ZOH), and TMS32010), and plant. The ZOH is normally contained within of the D/A convertor.

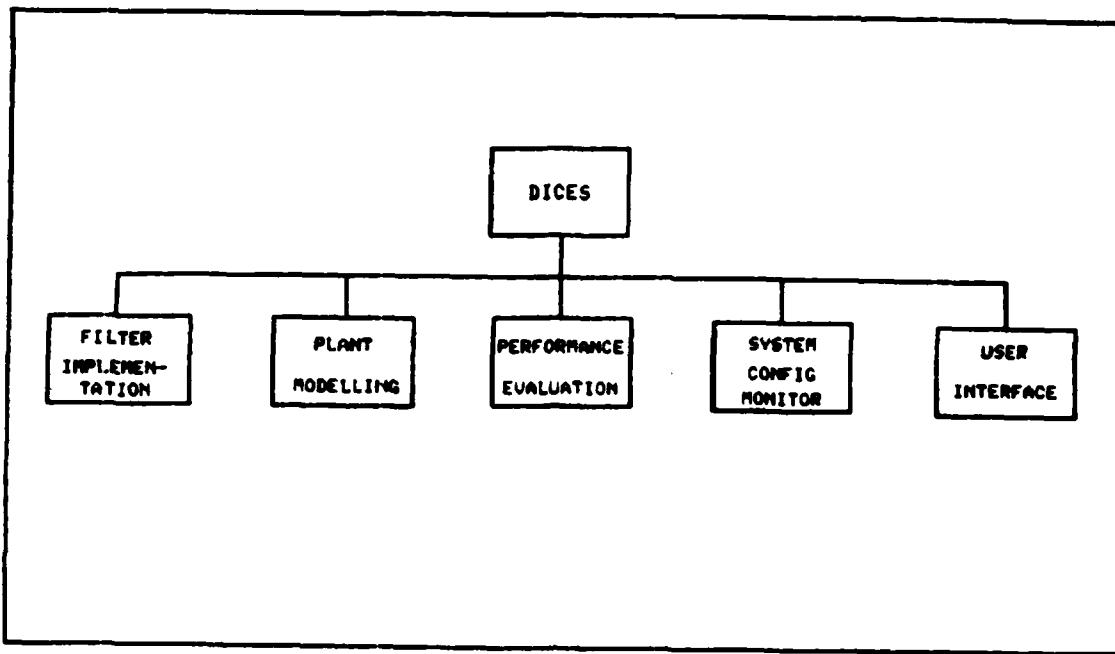


Figure 3. Functional Areas

Literature Search

Current and past literature reviewed included work in the area of hybrid computer simulations (11,13), design of Automated Test Systems (40), implementation of digital filter designs (3,4,6,8,13), effects of finite word length implementations (6,8,13), and analog computer simulation techniques (35).

Scope of Thesis Effort

During Phase One, a Data Flow Diagram technique (see appendix C) is used to analyze and define system requirements. This is accomplished by a top-level functional decomposition which allows detailed analysis of the system functions. These requirements are then allocated to functional blocks within DICES.

Phase Two consists of the design and implementation of a subset of requirements from Phase One. This system is then tested to determine compliance with the requirements levied upon it.

Recommendations for further enhancements to DICES are given in order to increase its usefulness and versatility as a design tool.

Structure

This thesis is organized in accordance with the general sequence of activities undertaken during this investigation, keeping in mind that the development process is an incremental, iterative process. This chapter presented an introduction to the topic of this investigation and the background of the topic.

Chapter II defines the system design requirements followed by the system functional requirements. The requirements are then analyzed to determine the proper functional block to which they will be allocated.

Chapter III designs a software and/or hardware system that, when implemented, will meet the requirements levied upon each functional block of the system.

Chapter IV describes the implementation via software and/or hardware of the design from Chapter III.

Chapter V describes the tests performed on the functioning system and presents the procedure used to implement the tests.

Chapter VI discusses the conclusions and recommendations from this investigation.

Appendices are:

A - DICES Users Manual.

B - Data Dictionary, consisting of

- 1) Subroutines
- 2) Data element/data flow descriptions
- 3) Process descriptions
- 4) File descriptions

C - briefly describes Data Flow Diagrams

D - System Test Data

E - VAX FORTRAN program listings which make up the host-resident software portion of DICES

II. Requirements Analysis and Definition

Introduction

In order to design a system of any complexity, it is essential to fully understand the requirements of the system, i.e., what is the system supposed to do and how well should it do it? It is also imperative that a structured, top-down flow of requirements takes place in order to be certain that the final system design meets each of the requirements.

It is the intention of this chapter to first define the system design requirements followed by the system functional requirements. The requirements will then be analyzed to determine the specific functional block they will be allocated to within the system. Implementation of the requirements allocated to each functional block will be accomplished in chapter four by a hardware and/or software module designed to satisfy the requirements.

System Requirements

A general block diagram of DICES as viewed from a CAD perspective is shown in figure 4. There is a user interface (I/F), control, data base, external interface, and filter hardware (H/W) (the digital controller/plant system).

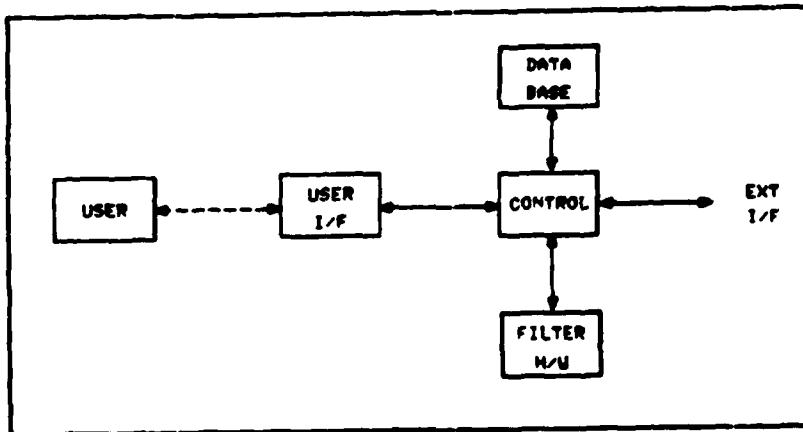


Figure 4. CAD Perspective of DICES

The top-level system design requirements are as follows (16):

- o Efficient User Interface (15:130)
- o Modularity
- o Concurrent Documentation (17:32)

Each of these system requirements (see figure 5) are now discussed in more detail.

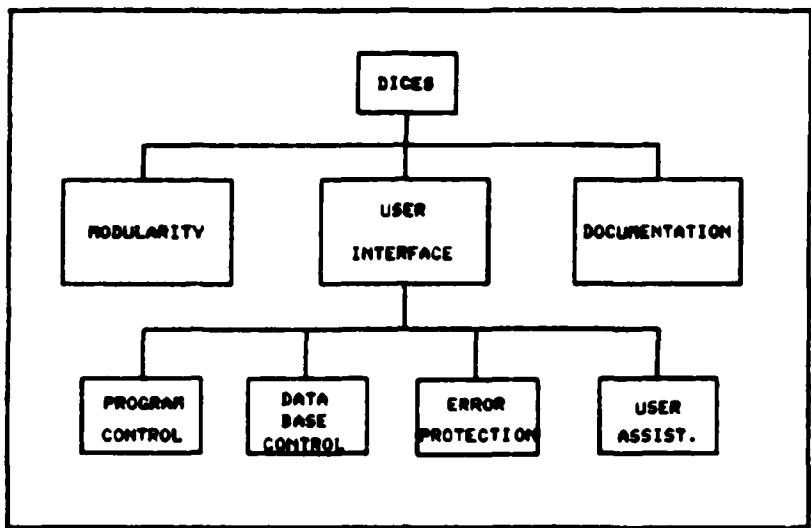


Figure 5. System Design Requirements

User Interface

Man-machine interaction or communication is a two-way information exchange to accomplish a specific objective. This communication must actively involve both parties: The user provides creativity and decisions while the computer provides fast and accurate storage and retrieval of data, and performs rapid calculations (18:3). An interactive mode must be used in order to provide feedback to the user as to the status and response to input. The user can direct the system to provide assistance (HELP files), correct an

input, or perform some other system task (17:13).

According to Larimer (15:14), an effective interactive user interface consists of program control, data base control, error protection, and user assistance.

Program Control (16:13-14). The user must be given a high degree of flexibility and control over the system operation. The user must be able to:

- o Stop the system and later restart without loss of data
- o Abort a command without terminating the entire simulation
- o Control the flow of the system's actions with as little effort as possible

In summary, the user must have nearly complete control of the system's operations and sequence of activities. The system must however maintain a watch over the user input to avoid 'dumb' mistakes.

Data Base Control. According to Weinberg (19:313), a data base is a file of interrelated data that are stored together to serve one or more applications and are independent of the programs using the data. In this application, the data base will consist of user designed

filter coefficients and performance test results. The user must be able to :

- o Retrieve the design parameters from a previous session and use them again
- o Store current work for later use
- o Store and retrieve test results for later comparison

Error Protection (16:15-16). DICES must provide some form of error protection and recovery to avoid making DICES attempt to execute an illegal operation (either in a FORTRAN sense or an operation). According to Thames (20), a CAD system must be able to tell the user that it understood what the user directs, and more importantly, when the system does not. The system must provide :

- o Protection against premature program termination due to user input errors
- o Ability to recover from input errors without starting from 'scratch'
- o Good error messages and other forms of user feedback

User Assistance - HELP (16:16). Because both new and experienced users may use the system, the user interface must cater to both. The novice may require substantial amounts of assistance, while the experienced user may need none. The HELP function must provide:

- o HELP to the user at any time when the next step is not understood by the user
- o Ability to selectively list options to the user

Modularity

Modularity in a relatively large system such as DICES is mandatory. Furthermore, the system is modular by nature, just because of the 'clearly defined' requirements, e.g., filter implementation, performance testing. This enables the large system to be decomposed into several smaller, more manageable systems. The modules of the system should exhibit a low degree of 'coupling' and a large degree of 'cohesion' to obtain some measure of independence of each module (6:423-472). According to Kernighan and Plaugher (21:64), modularity of design is best applied to the system design when modules are loosely coupled (highly independent) and each module 'does one thing and does it well' (functionally cohesive). Clearly, this approach makes sense for almost any system, be it software, hardware, or a combination of both.

Documentation

Documentation is defined as information about a system

available in writing (22:385). Documentation of the entire development process is required in order to document decisions made during the requirements definition, design, and implementation phases. While it is important to document hardware thoroughly, it is especially important to document software. Software exists only in its documentation (6:424), so if a poor job is done in documenting a software system, then the product is a poor software system, whether it currently works or not. The major item of user documentation for DICES is the DICES User's Manual. This manual will present an informal introduction and overview of the system and information about what problems can be solved by the system.

In addition to the user's manual, there will be detailed documentation describing the design process and the implementation of the design requirements. The software will be documented both in comment form within in the code as well as detailed documentation (flowcharts, data dictionaries, modules descriptions) within this thesis.

Grogono (23:351) proposes several guidelines for 'in-code' documentation as outlined in Logan's work (16:19):

- o The program should include a prologue with a brief title for the program, the name of the programmer, a description of what the program does, and a description of the input required by the program as well as output produced.

- o If complicated algorithms are used, provide references to the sources of the algorithms.
- o Document each procedure used, i.e., describe what it does, how it does it, and describe the role of each parameter.
- o Clarify potentially obscure sections of code with comments.
- o Document any parts of the program that might not be machine independent.
- o Historical data

Functional Analysis and Requirements Definition

DICES must be capable of accepting inputs from the user such as commands, controller algorithms, output instructions, and performance test instructions. These instructions are ultimately used to implement the controller design on a microprocessor system. After closing the loop around the plant, DICES must be able to determine how well the controller/plant system performed. These results should then be made available to the user to

allow comparison of other designs or with the theoretical results obtained from a CAD package. The functional flow of the 'control problem' (5:15) and how DICES relates to that flow follows. The control problem as mentioned above was examined and broken into six major processes (Fig 6). DICES accomplishes the last four of these processes to varying degrees, while the first two are performed prior to using DICES. A top-level description of each process level is given followed by an in-depth analysis of each level using Data Flow Diagrams (DFD) (6:427-433). This top-down decomposition approach is a graphical technique that is organized as a tree structure with a parent-child relationship (6:431). While this technique is described in the reference as a software engineering tool, it is very well suited for a system design project such as DICES. A short summary of the DFD technique is given in Appendix C.

Top-Level Functional Flow Description

Process 1.0, establishing the performance specifications of the system under study, is the first step in solving a control problem. There are many trade-offs here and the original performance requirements may eventually be modified if they are not attainable within other constraints (e.g., cost, schedule, technical).

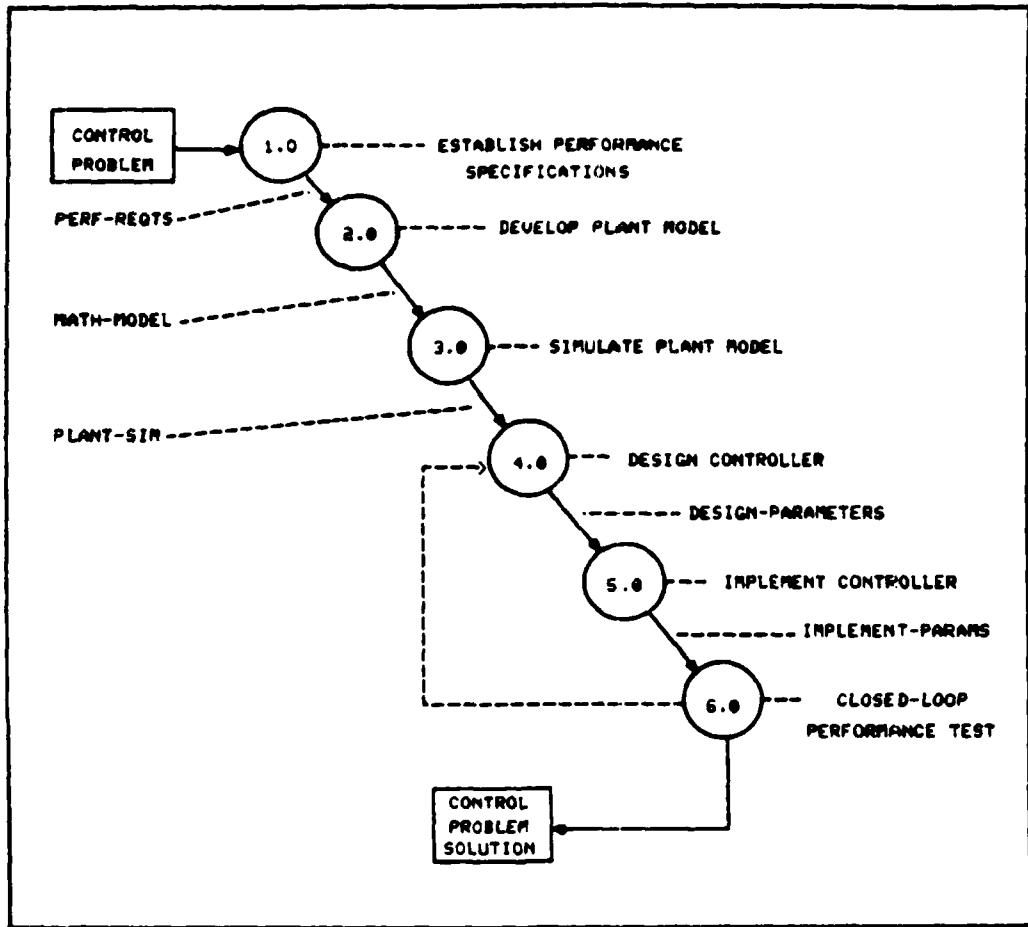


Figure 6. Top-level Functional Flow

Process 2.0, the development of a plant model, is typically the critical factor in designing a useful, practical control algorithm (12:25). If a good mathematical model is not obtained, then the design effort that follows will probably be wasted! This is also probably the most difficult step in designing a good controller. The designer typically has many trade-offs to make during the model development phase, such as linear vs. non-linear differential equations, stochastic vs. deterministic, and reduced-order vs. higher-order, higher fidelity model. The challenge is to develop the simplest model that is adequate to allow the design of a controller algorithm that meets the system specifications.

Process 3.0, the simulation of the plant model, is the first step in which DICES plays a role. Processes 1.0 and 2.0 are done prior to using DICES (5,6,12,13,14). In DICES, the simulation of the SISO plant is done using conventional analog computer techniques on a EAI TR-48 Analog Computer System. The user can develop as complex a model as desired. DICES will assist the user in interfacing the plant simulator to DICES, instructing the user how to connect cables and so forth.

Process 4.0, the design of the controller is performed using a CAD package such as TOTAL or ICECAP (15,16). DICES will allow the user to 'connect' to the system containing the CAD packages and the user is free to design the

controller. When the analysis indicates an adequate design, the controller transfer function is then transferred in factored form to the next process step.

Process 5.0, the implementation of the controller transfer function, $G_{hc}(Z)$ is performed by using the factored form of $G_{hc}(Z)$ to implement $\lfloor(n+1)/2\rfloor^*$ cascaded second-order sections. The basic structure of the second-order sections (1D, 2D, etc) (6,13) is an option the user determines. This implementation is accomplished by determining the coefficients and scaling factors to be inserted into a general-purpose cascaded second-order digital filter program that is loaded into the TMS32010 microprocessor.

Process 6.0, closed-loop performance testing, is accomplished by inserting the controller into the closed-loop control system and executing the controller algorithm. The tests to be performed on the closed-loop system are remotely programmed into the B/K 2032 System Analyzer by the VAX 11/780 via the IEEE-488 bus. The B/K 2032 monitors the input and output of the closed-loop system. The test sequence is executed until completion and

* = smallest integer greater than or equal to $(n+1)/2$

the results are displayed on the user's terminal or on the screen of the system analyzer. The user also has full front panel control of the B/K 2032 System Analyzer which permits analysis of the test data. The output may also be directed to a line printer or X-Y plotter.

After the test process is complete and results stored, the user can compare the results with the theoretical results and determine if the design iteration shown in figure 4 is required. If not, the controller is complete and ready to be used to control the real plant.

Functional Areas

The process levels divide DICES naturally into five major functional areas. These functional blocks will be allocated specifications at the completion of the requirements definition phase. The five functional blocks of DICES are:

- a. User interface - Provides the user with menu-driven options, data storage, and results of tests.
- b. System Monitor and Configuration - Provides the control necessary to configure components of system for initialization, test, etc.
- c. Filter Implementation - Provides the implementation of the digital controller design, $G_{hc}(z)$, to be executed on the microprocessor.

d. Plant Modelling - Provides the user with an analog computer simulation of a physical plant or process to be controlled.

e. Performance evaluation - Controls the performance of the tests on the System-Under-Test (SUT), which includes controller and plant.

Now that the top-level functional flow and system functions have been addressed, it is necessary to analyze each of the last four process levels individually in more detail. Each will be analyzed and requirements defined and allocated to a major functional block of the system as described in chapter 1. Process levels 1.0 and 2.0 will not be discussed in any more detail as references abound that address the basic 'control problem' (5,6,12,13,14).

Detailed Functional Analysis

Level 1.0 - Establish System Performance Specifications

(See References 5,6,12,13,14)

Level 2.0 - Develop Plant Mathematical Model

(See References 5,6,12,13,14)

Level 3.0 - Simulation of Plant Model

In order to perform a hybrid simulation, the plant

must be simulated using conventional analog computer techniques. This entails implementing the differential equations developed in process 2.0 on an analog computer.

While DICES does not directly perform the simulation for the user, it does provide some guidelines and assistance.

Level 3.0 consists of 4 sub-processes (See figure 7). The input to the process is a mathematical model that describes the dynamic system under study. These equations describing the system can be of arbitrary complexity, however, they must be implementable on the available analog computer. The output of process 3.0 is a hardware simulation on an analog computer that attempts to faithfully mimic the plant. The user wishes, of course, to alter the plant's dynamic response. This simulation can be linear or non-linear, depending on the desired complexity and design techniques used.

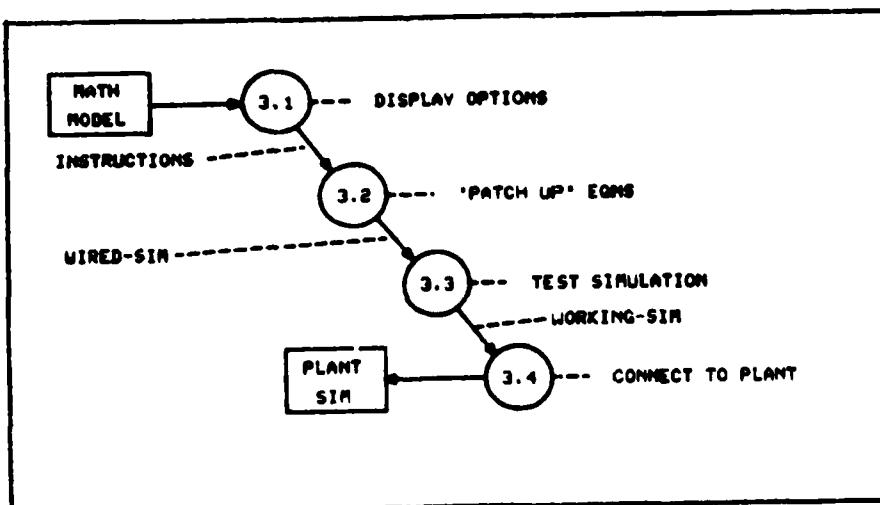


Figure 7. Level 3.0 - Simulate Plant Model

Level 3.1 - Display Simulation Options

This process must present the user with the various options available. These consist of

1. Display of electrical and mechanical interfaces the user must connect.
2. Present step-by-step instructions to configure the system.
3. Inform user as to further reference documentation.

Level 3.2 - 'Patch Up' Equations

This step involves having the user actually wire the analog computer ('patching up') to perform the various integrations, multiplies, and summations that must be performed to solve the differential equations of the system. The details of this process are beyond the scope of this work, however, there are many excellent references on this subject (28,35).

Level 3.3 - Test Simulation Model

This process consists of stimulating the plant model and observing the open and/or closed-loop response of the system. This response can then be compared with the results obtained from a CAD package to determine how well the results match the analytical results. The adequacy of the plant simulation should be ascertained before continuing with the simulation, as the entire simulation depends upon an accurate, working plant model.

Level 3.4 - Connect Controller to Closed-Loop System

This process consists of connecting the digital controller into the closed-loop system. This will place the controller into either the forward or feedback loop of the closed-loop system. The analog computer must, of course, be compatible electrically with the TMS32010 AIB or any other A/D and D/A converter used. The analog computer should also have standard interfaces to allow easy, fast connections to the AIB.

Level 4.0 - Design Controller

After the plant has been modelled to the user's satisfaction, it is then necessary to design (See figure 8) the compensator, controller, or digital filter (all of which are used interchangeably). This controller is typically referred to as a cascade or feedback compensator or controller. The purpose of this controller is to alter the dynamic response of the plant. This controller may range from a simple proportional cascade controller to a complex, high-order feedback controller (5,6,12,13,14).

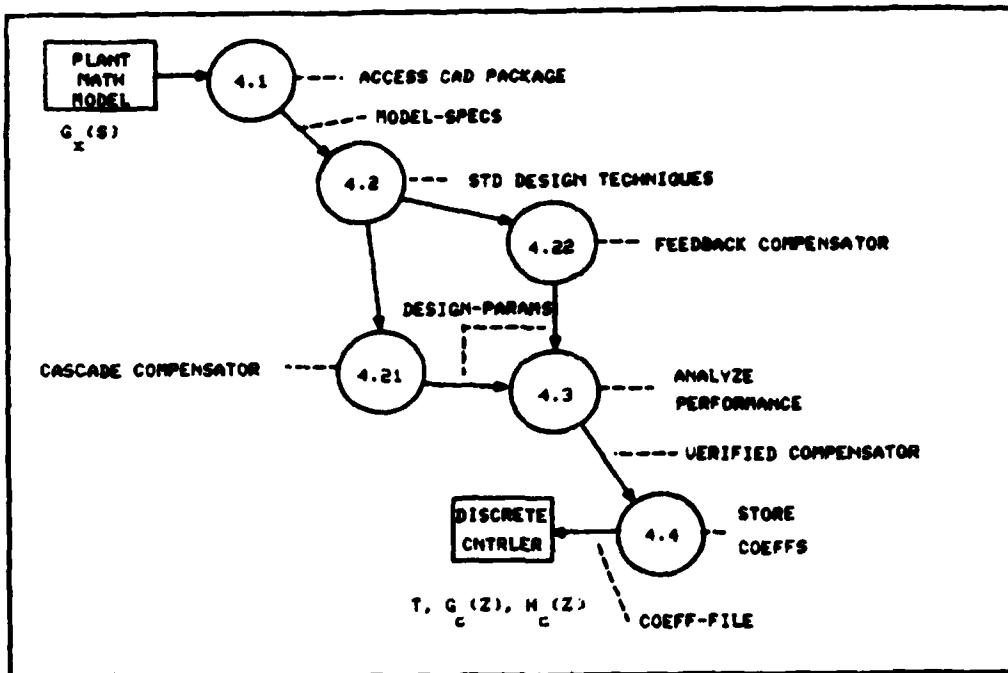


Figure 8. Level 4.0 - Design Controller

There are many CAD packages available to assist the designer in this phase (15,16). In the case of a digital controller design, there are two primary methods, the Digitization (DIG) and the Direct (DIR) method (6:354-381).

The input to this process is the mathematical plant model developed in process 2.0. This model is the basis for all the design steps that follow in this process. The output of this process is a discrete controller design with

quantized filter weights (coefficients of a polynomial in Z) that meets the specifications set forth in process 1.0 (in a theoretical sense). The sample period, T, is also output. These filter weights and value of T are stored in a data file in factored form for later scaling, ordering, and pole-zero pairing (13).

Level 4.1 - Access CAD Package

This process establishes communication for user interface with a CAD package that allows system analysis and design. The local DICES computer should appear transparent while in this mode of operation. The user follows the protocol of the operating system under which the CAD package is operating. The DICES terminal accepts and should display all CAD package input and output.

Level 4.2 - Standard Design Techniques

This process involves using any of the standard design techniques available to design the controller. The details of this step are beyond the scope of this thesis effort. There are many excellent references on this subject (5,6,13,14). Figure 9 gives an idea of the many approaches that can be taken to design an adequate controller (6:192-193).

The output of this step is often a feedback or cascade controller of order less than eight.

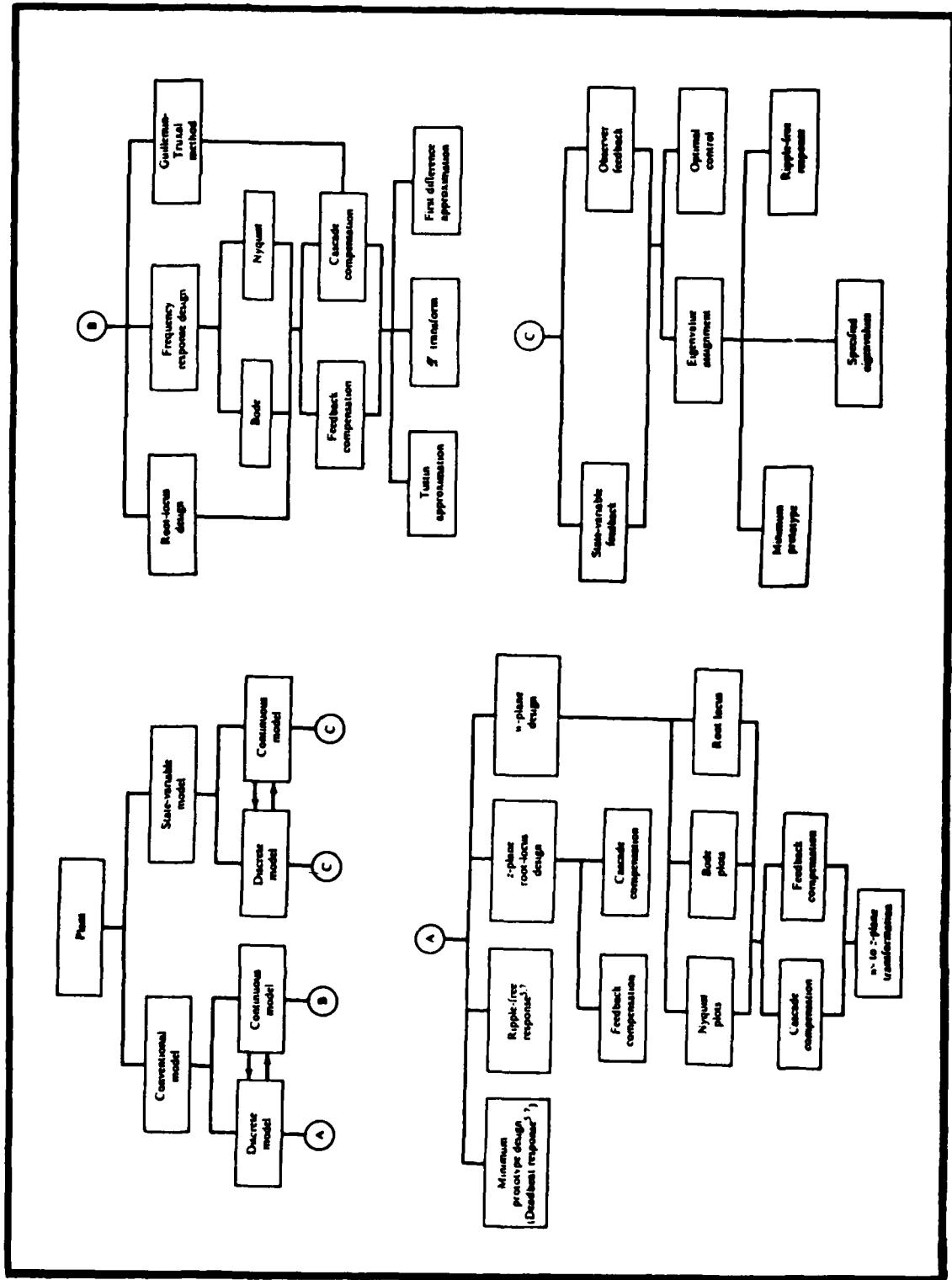


Figure 9. Analysis and Design Methods (6:192-193)

Level 4.3 - Analyze System Performance

This process generally involves placing the controller in a closed-loop cascade or feedback configuration and analyzing the system performance from a theoretical viewpoint. This may involve obtaining frequency response (magnitude and phase) plots, impulse, and step response information about the closed-loop system.

Level 4.4 - Save Controller Coefficients

This process saves the coefficients that emerge from the previous steps into a file that can be accessed at a later time. The files should contain the coefficients of the factored form of the controller transfer function.

Level 5.0 - Implement Controller

Once the controller is designed to the user's satisfaction and the design meets the system specifications theoretically, it is necessary to implement the design using hardware and software (See figure 10). The input to this process is the factored form of the controller, $G_{hc}(Z)$ and the sample period, T . The controller is in general of order n , and the numerator is of equal or less order. The process pairs the poles and zeros for implementation as second-order cascade sections and also computes the scale factors needed to avoid arithmetic overflow. The modules

are then ordered and the code for the TMS32010 is generated and loaded into the EVM. The output of this process is code that when executed on the EVM, will implement the desired controller difference equation.

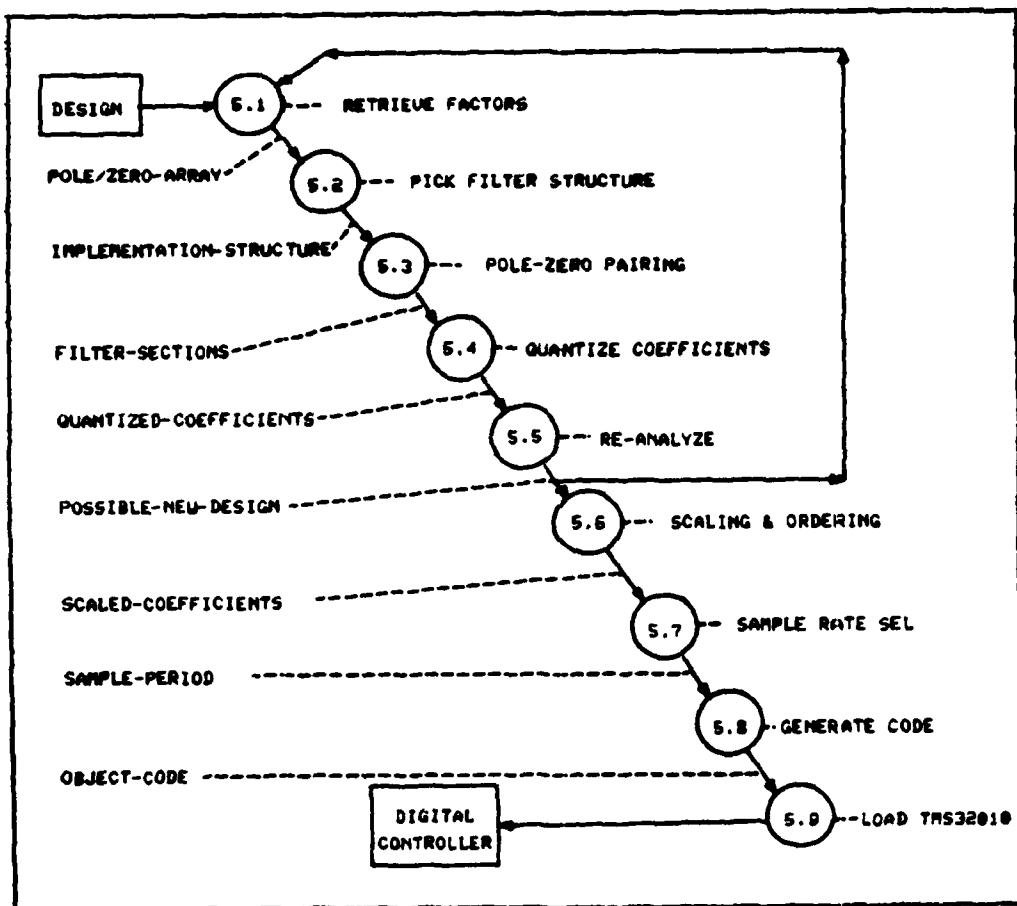


Figure 10. Level 5.0 - Implement Controller

Level 5.1 - Retrieve factors

This process establishes communication with the device that has the controller coefficients ($b_0 \dots b_n, a_1 \dots a_{n-1}$) stored in factored form, i.e.,

$$G_{hc}(Z) = \frac{b_0(Z + b_1)(Z + b_2)\dots(Z + b_m)}{(Z + a_0)(Z + a_1)\dots(Z + a_{n-1})} \quad (1)$$

This process also loads the coefficients into the workspace of the local DICES controller.

Level 5.2 - Pick Filter Structure

This process presents the standard filter implementation structures to the user who then selects the structure desired (6:332-335, 590, 13:355-379). These standard structures are commonly referred to as the 1D, 2D, 3D, 4D, 1X, and 2X structures. Each structure has certain time and/or space advantages. The direct structures (1D, 2D, 3D, and 4D) have the difference equation coefficients explicitly used in the implementation of the filter, while the 1X and 2X structure have real and imaginary components as coefficients in the structure. The 1X and 2X structures are referred to as cross-coupled structures and are used to realize complex poles.

Generally, the approach taken to realize a high order filter is to cascade second-order sections, each of which can be implemented with one of the above standard structures (direct or cross-coupled). The cascade second-order approach is used to reduce coefficient quantization sensitivity. This stems from the fact that as the order of a polynomial increases (e.g., a high-order direct implementation), the roots become more sensitive to coefficient perturbations. In a digital filter, these perturbations are due to the quantization of the coefficients.

The 3D and 4D structures are non-canonical because they contain more than the minimum number of time delay elements (n in this case). The other four structures are of a canonical form (6:332, 13:357).

When implementing the digital filter in hardware, there are more distinct advantages of one form over another. However, when using software/hardware to implement the filter, the advantages of one structure over another become less discernible. This is mainly because the number of summing junctions and signal distribution points are critical in a hardware implementation, but only imply additional software operations.

Level 5.3 - Pole-Zero Pairing

This process analyzes the zeros and poles of the compensator and selects optimum or 'near optimum' pairings of two zeros and two poles in order to build a second-order section (6:338,590). This pairing process continues until all the zeros and poles have been paired. If the order of the filter is odd (i.e., n is odd), then one second order section will have some zero coefficients in order to realize a first-order section.

When the coefficients are quantized, pole-zero migration occurs as discussed in process 5.2. This phenomena is somewhat analogous to component tolerances in an analog filter implementation. Although minimized because of the second-order nature of the section, some movement still occurs. The sensitivity of each root with respect to each coefficient can be determined and used as a guideline to assist in solving problems involving truncating versus rounding of certain coefficients. The sensitivity of each root (in a second-order section) is inversely proportional to the distance the roots are located from each other in the Z-plane. Therefore, to minimize the effects of coefficient quantization, the poles and zeros are combined using an 'optimal' pairing algorithm (6:338,590, 13:481).

Level 5.4 - Quantize coefficients

This process consists of quantizing the coefficients into a 16-bit word and re-analyzing the performance of the closed-loop system.

Because of the finite wordlength available to represent an infinite number of possible coefficients, there are errors generated when this quantization occurs. Obviously the larger the wordlength, the less error generated when the coefficients are quantized. This quantization causes pole-zero migration in the Z-plane (6: 336-339). The sensitivity of each root of the polynomial in Z can be calculated with respect to each coefficient. This gives the designer added information with which to manipulate the most influential coefficients.

Level 5.5 - Re-Analyze System Performance

After the coefficients are quantized to 16 bits, it is necessary to re-analyze the closed-loop response of the system with the 'new' poles and zeros. If stability and performance are still acceptable, the user can proceed to the next step in the process. If not, the user must go back to process 5.4 and use a different method of quantizing the coefficients, e.g., truncating instead of rounding the most influential coefficients.

Level 5.6 - Scaling and Ordering

The objective of this process is to select a scale factor for each independent second-order module in order to prevent arithmetic overflow from occurring within the module. It also orders the modules to minimize the effects of quantization. Scaling attempts to allow the use of the full dynamic range of the fixed-point number representation. This helps improve the signal-to-noise ratio by keeping the signal level near the maximum number representation while simultaneously preventing overflow. Overflow is a very serious problem and should be avoided at all times! The effects of arithmetic overflow and 'wrap-around' are a form of limit cycles called 'overflow oscillations' (6:343-345, 13:462).

This process orders the first and second-order sections in the cascade structure. The ordering of the modules serves to reduce the output noise and limit cycle response. There have been many ordering algorithms proposed in the literature (24-27).

Level 5.7 - Sample Rate Selection

This process determines the minimum sample rate that should be used to obtain acceptable results from the digital control system. The user should have the option to override the selection made by DICES if desired.

Level 5.8 - Generate Code

This process takes the specifications of the previous processes and translates the design into a program that when assembled, loaded, and executed on the TMS32010 microprocessor will implement the controller. The code to be generated will consist of a generic filter program which will allow implementation of first and second order filter sections. This step will fill in the particular coefficients and scaling factors to be implemented for the particular controller under development. After insertion of the above parameters, the program will be loaded into the VAX 11/780 for assembly using the TMS32010 Assembler, and then downloaded to the TMS32010 microprocessor.

Level 5.9 - Load TMS32010

This process will take the object code generated from the last process and load the TMS32010 program memory. This load will take place under control of the user and can be initiated and terminated from the user terminal.

Level 6.0 - Closed-Loop Performance Testing

After the design and coding is complete, the controller is inserted into the closed-loop system and tested to determine its performance with the simulated

plant. The input to this process is the completed controller ready to be inserted into the closed-loop system. This process queries the user for test options and test parameters. It then commands the programmable test instrumentation and initiates the tests. The results are then returned to the user when complete.

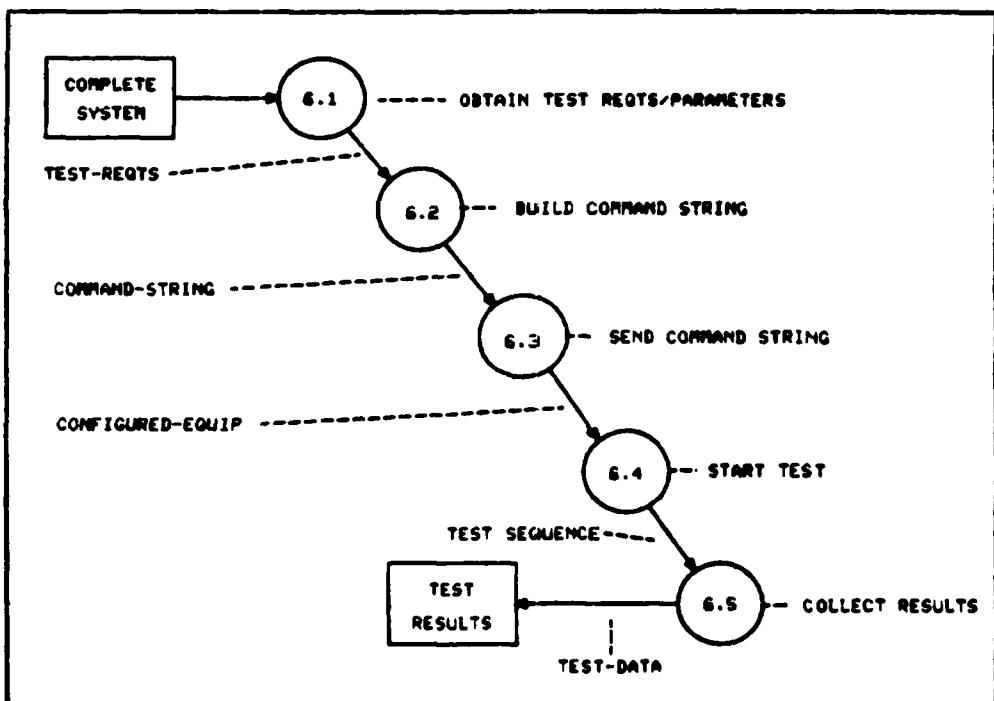


Figure 11. Level 6.0 - Closed-Loop Performance Testing

Level 6.1 - Obtain Test Requirements and Parameters

This process queries the user for tests to be performed on the closed-loop system and also obtains the test parameters associated with the tests being requested. The user has the choice of the following tests to be performed on the closed-loop system:

- o Closed-loop frequency response (magnitude)
- o Closed-loop frequency response (phase)
- o Impulse response
- o Step response

Default settings for the various test equipment are also written to each item used in the system. These settings are stored within the test equipment for later use.

Level 6.2 - Build Command String

This process assembles the proper command string for later transfer to the programmable test instrumentation. This command string contains all the information required by the test instrumentation to perform the required tests. This command string is typically an American Standard Code for Information Interchange (ASCII) character string that is transmitted over an interface bus using a standard protocol, such as the IEEE-488 standard (38, 39).

Level 6.3 - Send Command String

This process transmits the command string to the programmable test instrumentation. This process will typically invoke a device interface driver which handles the detailed handshaking and electrical interface requirements needed to communicate over a bus such as the IEEE-488 standard. DICES will assemble the command string required to be sent and pass it to a machine language subroutine which will perform the hardware/software interface function with the IEEE-488 interface module within the DICES computer.

Level 6.4 - Initiate Test

This process commands the test instrumentation to begin the actual programmed test sequence. This is simply a start command to the test equipment.

Level 6.5 - Receive Test Results

This process monitors the status of the on-going tests and receives the test results upon completion. This process then displays a summary of the test results to the user on the video terminal and presents options for a more detailed presentation of the test results.

Functional Requirements Allocation

The requirements that have been defined in the previous sections must now be allocated to the five major functional area of DICES. Each requirement will later be allocated within these areas to sub-functional areas to be implemented in hardware and/or software.

Level 1.0 - Establish System Perform Specifications

All of level 1.0 is allocated to the User Interface Module.

Level 2.0 - Develop Plant Mathematical Models

All of level 2.0 is allocated to the User Interface Module.

Level 3.0 - Simulation of Plant Model

The following table indicates the allocations within this level. Additional allocations follow the table where a requirement may be allocated to different modules:

REQ	FILTER IMPLEMENT	PLANT MODEL	PERF EVAL	SYSTEM CONFIG	USER I/F
DISPLAY OPTIONS (3.1)					X
PATCH EQUATIONS (3.2)		X			
TEST SIM (3.3)					X
CONNECT SYSTEM (3.4)		X		X	X

Table 1. - Level 3.0 Requirements Allocation

Level 3.2 - Patch Up Equations

The User Interface Module is allocated the requirement of assisting the user in planning the analog simulation.

The Plant Modelling Module is allocated the requirement of providing the hardware necessary to implement the plant model.

Level 4.0 - Design Controller

The following table indicates the allocations within this level:

REQT	FUNCTIONAL AREA				
	FILTER IMPLEMENT	PLANT MODEL	PERF EVAL	SYSTEM CONFIG	USER I/F
ACCESS				X	
CAD (4.1)					
DESIGN CONTROLLER (4.2)					X
ANALYZE					X
PERF (4.3)					
STORE COEFFS (4.4)					X

Table 2 - Level 4.0 Requirements Allocation

Level 5.0 - Implement Controller

The following table indicates the allocations within this level:

REQT	FUNCTIONAL AREAS				
	FILTER IMPLEMENT	PLANT MODEL	PERF EVAL	SYSTEM CONFIG	USER I/F
RETRIEVE					X
FACTORS (5.1)					
SELECT					
STRUCTURE (5.2)	X				
POLE/ZERO					
PAIRING (5.3)	X				
QUANTIZE					
COEFFICIENTS (5.4)	X				
RE-ANALYZE (5.5)					X
SCALING & ORDERING (5.6)	X				
SAMPLE RATE SELECTION (5.7)	X				
GENERATE CODE (5.8)	X				
LOAD CODE (5.9)				X	

Table 3 - Level 5.0 Requirements Allocation

Level 6.0 - Performance Testing

The following table indicates the allocations within this level:

REQT	FUNCTIONAL AREAS				
	FILTER IMPLEMENT	PLANT MODEL	PERF EVAL	SYSTEM CONFIG	USER I/F
TEST REQTS (6.1)					X
CMD STRING (6.2)			X		
SEND CMD (6.3)			X		
START TEST (6.4)			X		
COLLECT RESULTS (6.5)			X		X

Table 4 - Level 6.0 Requirements Allocation

III. System Design

Introduction

This chapter describes the design DICES to meet the requirements levied upon it in the previous chapter. It describes the design of each major module and discusses the overall system.

System Design Constraints

There are several design constraints imposed on the development of DICES. These are mainly constraints on the use of certain hardware items that are available in the Information Sciences Laboratory. These constraints are not to be viewed as limitations on the design of DICES, for some of these constraints are actually the best choice that could have been made even if this were a completely unconstrained project.

EVM/AIB Board

The first of the system level design constraints is the requirement to use the TMS32010 microcomputer EVM and AIB modules that are available in the laboratory. It has been shown (13) that the TMS32010 would probably be selected over most of the other available DSP microcomputers and almost certainly over the general-

purpose microcomputers that are not optimized for DSP algorithms.

Computer System

The next constraint is the use of a computer system for executing the DICES program and controlling the system analyzer. The options were a Plessey LSI-11, VAX 11/730, and a VAX 11/780. A trade-off analysis was performed to determine the appropriate computer system or mix of computer systems to use for DICES.

A summary of the trade-off matrix is shown on the next page. The ratings in each category are Excellent (E), Good (G), Poor (P), and Very Poor (VP). Some of the categories do not lend themselves to a clear-cut determination of the rating or comparison, however, it is felt that the matrix represents a fair comparison of the three systems to accomplish DICES' objectives.

<u>Category</u>	<u>LSI-11</u>	<u>11/730</u>	<u>11/780</u>
<u>Software</u>			
HOL	G ¹	E	E
IEEE-488 S/W	G ²	E	E
Future S/W Expansion	G	E	E
Operating System	G	E	E
I/F with existing Controls S/W	P	G	E
Support S/W Avail	G	G	E
S/W Maint. Contract	NO	YES	YES
<u>Hardware</u>			
Speed	P ³	G	E
IEEE-488 H/W	E	P ⁴	E
Future H/W Expansion	G	E	E
MODEM	NO	NO	YES
Mobility	E	P	VP
H/W Maint. Contract	NO	YES	YES

Notes:

- 1 - FORTRAN IV. No character or bit manipulations
C Language available
- 2 - Hewlett-Packard Basic available
- 3 - Compile/link time very slow
- 4 - Presently no room for additional H/W

Table 5. Trade-off Results

Trade-off Summary

Without assigning numerical values to the rating system, it is obvious that the VAX 11/780 should be used to implement as much of DICES as possible. An effort should be made to use only the VAX 11/780 if possible. DICES can be transferred to the VAX 11/730 at a later date if desired. The LSI-11 should not be considered as the computer system for DICES.

System Component Description

TMS32010 EVM/AIB

The TMS32010 is a Digital-Signal-Processing (DSP) microprocessor introduced by Texas Instruments, Inc. in 1983. Its architecture, speed, and instruction set are designed for efficient execution of digital signal processing algorithms. The table below summarizes its main features and performance characteristics (2,3).

Configuration: Microprocessor version - TMS32010.

Clock/Instruction Cycle: 20 Mhz clock/200 nsec instruction cycle. On-chip oscillator.

Memory: 144 Words of on-chip data RAM.
4096 words of external program ROM.

Computational Sections: Double-Precision 32-bit ALU and Accum.
200 nsec multiplier.
Overflow mode that saturates the accumulator on arithmetic overflow.

**Program Control
Section:** 16-bit bus for off-chip instructions.
 4x12 stack for context switching.
 Autoincrement/decrement registers.
 Polling pin for hardware interfacing.
 Vectored interrupt.

The TMS32010 use a modified Harvard architecture, (see figure 12), which permits the transfer of information between program and data memories. A strict Harvard architecture provides separate program and data memories with no cross transfer of data. This modified structure allows the designer the flexibility, for example, to reload the filter coefficients and the initial conditions of his control algorithm.

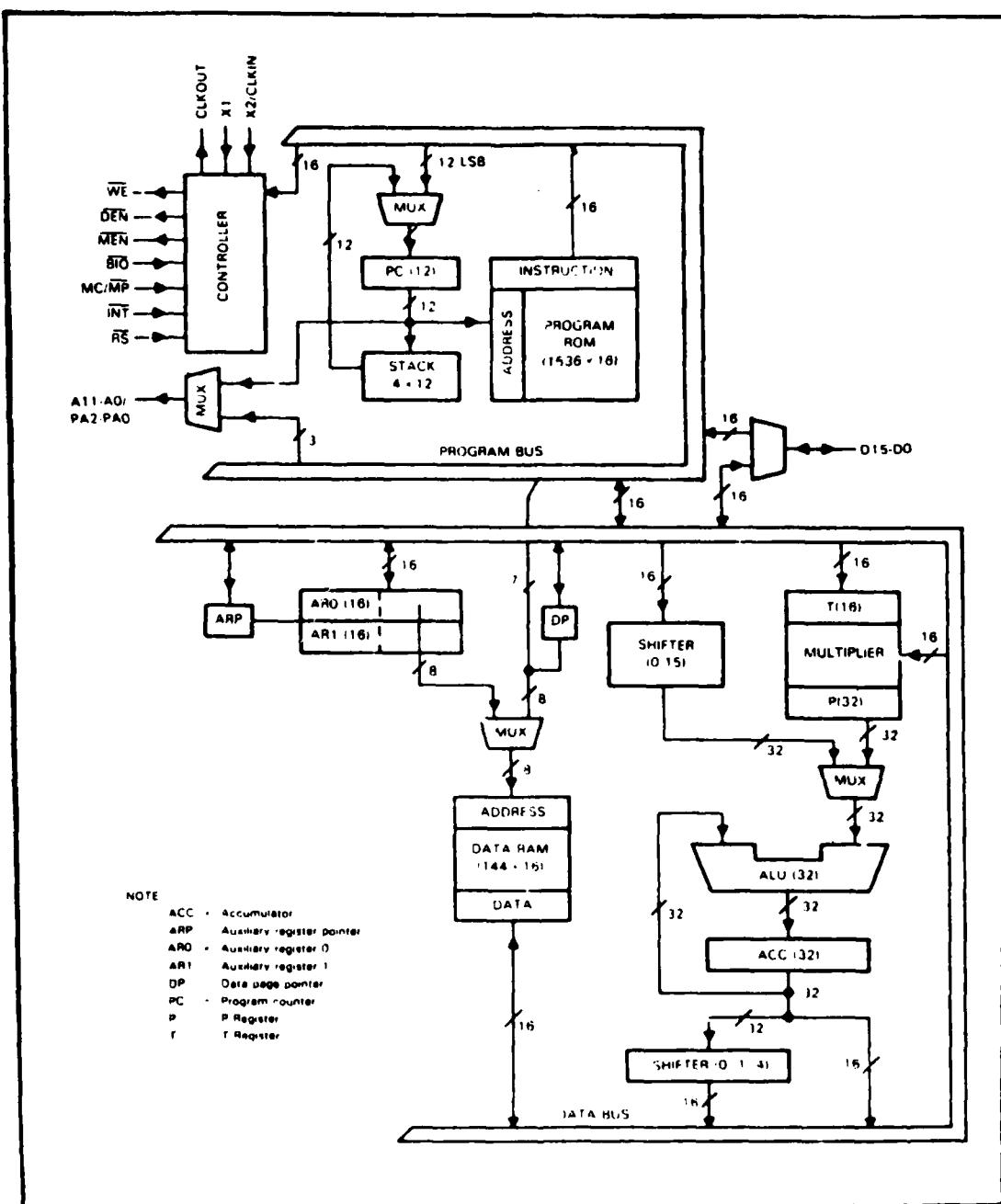


Figure 12. TMS32010 Architecture
(Reprinted from reference 3)

An outstanding feature is an on-board 16x16 bit multiplier, which produces a 32-bit product in one instruction cycle, i.e., 200 nsec. The TMS32010 also contains a barrel shifter that left shifts data 0 to 15 places before the data is transferred into the accumulator and a second shifter that shifts the upper half of the 32-bit accumulator 0, 1, or 4 places before it is stored in data RAM. This allows for the extraneous bits that are generated when using certain number representations within the TMS32010. The I/O structure consists of eight 16-bit input and eight 16-bit output ports.

The TMS32010 instruction set includes a set of special instructions necessary for fast implementation of sum-of-products computations encountered in digital filtering/compensation calculations. Most of the instructions critical to DSP execute in 200 nanoseconds. A typical multiplication of a delayed value by a coefficient stored in memory, and then a shift of the delayed data in memory executes in 400 nsec. The results of the previous operation is also accumulated in the accumulator during the same 400 nsecs. The TMS32010 has three addressing modes: direct, indirect, and immediate addressing.

System Development Tools

There are several system development tools that were used to support the development of the TMS32010 code used in this project. These tools are now described.

1. The TMS32010 Evaluation Module (EVM) is a development system for performance evaluation and design development. The EVM supports this design project very well because of its ease of use due to its on-board software development tools. The firmware package includes a debug monitor, editor, assembler, reverse assembler, EPROM programmer, communication software, and audio cassette interface. Some of these features were not used because of the extensive development software available on the Information Sciences Laboratory (ISL) Vax 11/780.
2. The TMS320 Analog Interface Board (AIB) is 12-bit analog-to-digital, digital-to-analog conversion board. The AIB contains anti-aliasing and smoothing filters (30:3-8). The basic sampling rate is programmable from 25 microseconds to 52.42 milliseconds. This range is later modified via software.
3. The TMS32010 Macro Assembler is a two-pass assembler which executes on the ISL Vax 11/780 and supports macro definitions and calls as well as conditional assembly. The assembler provides a comprehensive set of error diagnostics and symbol and cross-reference tables in the listing.
4. The TMS32010 Simulator is a program that simulates the TMS32010 on the ISL Vax 11/780 to allow the verification of assembly language programs.

There are other development tools which are available to support the TMS32010. Only the specific tools used in this thesis are mentioned above.

VAX 11/780

The configuration of the AFIT Information Sciences Laboratory VAX 11/780 at the time of this investigation is as follows:

Digital Equipment Corporation (DEC) VAX 11/780 processor with:

Virtual Memory Operating System - VMS Version 4.2
500K words of Physical Memory
Unibus

3 - DEC RA81 Disk Units
2 - DEC RK07 Disk Units
1 - DEC TU80 Magnetic Tape Unit
6 - DEC VT-240 terminals
7 - DEC VT-100 terminals
1 - Tektronix 4014 Graphics Terminal
1 - DEC LA-120 Operator Console
1 - Printronix high-speed line printer
1 - DEC Laser printer
1 - Hayes Smart-Modem

A National Instruments GPIB11-2 IEEE-488 Interface Card was also installed within the VAX 11/780 to permit remote programming of IEEE-488 compatible equipment.

B/K Model 2032 System Analyzer

The B/K 2032 is a dual-channel FFT system analyzer which can be used to analyze the performance of a dynamic system. It is fully programmable via a IEEE-488 interface and can also be completely controlled by its front panel.

The analyzer can determine the magnitude, phase, and impulse response of a system under test, to name just a few of its many capabilities. The frequency range is 0 to 25.6 Khz with 15 selectable frequency spans from 1.56 Hz to 25.6 Khz.

Other functions available include:

Instantaneous Time Function
Probability Density
Instantaneous Spectrum
Frequency Response
Signal-to-Noise Ratio
Auto Correlation
Impulse Response

EAI Analog Computer

The EAI TR-48 Analog Computer is a general purpose analog computer containing integrators, amplifiers, and potentiometers which can be used to model differential equations that represent physical dynamic systems (28). The linear amplifier voltage range is ± 10 volts.

Design Overview

The requirements that have been allocated to the five major functional areas must now be satisfied by a module within the appropriate area. Each requirement must be met by designing a hardware and/or software sub-system that,

when implemented, will meet the requirements levied upon it. Each major functional area will now be designed to meet each of those requirements.

The following tables show the grouping of each requirement with the five major modules. These are followed by the detailed design for each module.

User Interface

The User Interface Module (UIM) is the module which interfaces with the user and permits access to the functions that DICES has available.

The requirements are collected from the various levels and presented below.

<u>Level</u>	<u>Function</u>
1.0	Determine system performance specs
2.0	Develop plant model
3.1	Display options
3.3	Test simulation
3.4	Connect system
4.2	Design controller
4.3	Analyze performance
4.4	Store coefficients
5.1	Retrieve coefficients
5.5	Re-analyze performance
6.1	Determine test requirements
6.5	Collect test results

Table 6. UIM Requirements

Filter Implementation

The Filter Implementation Module (FIM) provides the resources needed to implement the controller design in TMS32010 code.

The requirements are collected from the various levels and presented below.

<u>Level</u>	<u>Function</u>
5.2	Select filter structure
5.3	Pole/zero pairing
5.4	Quantize Coefficients
5.6	Scaling and Ordering
5.7	Sample Period Selection
5.8	Generate TMS32010 code

Table 7. FIM Requirements

Plant Modelling

The Plant Modelling Module (PMM) involves wiring the analog computer system to produce the model of the plant and then actually connecting the plant simulation into the closed-loop control system.

The requirements are collected from the various levels and presented below.

<u>Level</u>	<u>Function</u>
3.2	'Patch-up' eqns on analog computer
3.4	Connect system together

Table 8. PMM Requirements

System Configuration

The System Configuration Module (SCM) assists the user in configuring the various system components. This module initializes equipment that requires start-up commands and also loads the TMS32010 code into the EVM for execution.

The requirements are collected from the various levels and presented below.

<u>Level</u>	<u>Function</u>
4.1	Access CAD package (ICECAP)
5.9	Load TMS32010 code
6.1	Default Parameters to Test Equipment

Table 9. SCM Requirements

Performance Evaluation

The Performance Evaluation Module (PEM) determines the user's test requirements and generates the appropriate commands necessary to program the remote test equipment.

The requirements are collected from the various levels and presented below.

<u>Level</u>	<u>Function</u>
6.2	Generate command string
6.3	Send command string
6.4	Start test sequence
6.5	Collect test results

Table 10. PEM Requirements

Detailed Module Design

Detailed User Interface Module Design

Level 1.0 - Determine System Performance Specs. This requirement shall be met by presenting a display which directs the user to analyze the system to determine the system performance specifications. As stated previously, this is done entirely by the user because of the many qualitative and quantitative factors that go into determining performance requirements to be placed upon a system.

This process may be rather obvious but is included for completeness in order to make DICES a complete system to transcend from the control problem to the control solution.

After execution of this process, the system shall return to the main opening menu.

Level 2.0 - Develop Plant Model. This requirement shall be met by presenting a display which directs the user to develop a mathematical model for the plant under study. This function is also done entirely by the user.

This process is also rather obvious but, like level 1.0, is included for completeness.

After execution of this process, the system shall return to the main opening menu.

Level 3.1 - Display User Options. This module shall display the options available to assist the user in interfacing the analog computer, digital controller, system analyzer, and other components of DICES.

The options shall be:

- 1) Step-by-step checklist
- 2) Further reference documentation

Option 1 shall present a step-by-step checklist to assist the user in connecting the system.

Option 2 shall present other documentation that the user can refer to in order to obtain more detailed information about some aspect of the interface phase of using DICES.

Level 3.3 - Test Plant Simulation. This module shall display instructions that advise the user to thoroughly test the plant simulation that is implemented on the analog

computer. This testing can be performed by stimulating the open or closed-loop response of the system and comparing it to that obtained by digital simulation (e.g., ICECAP, TOTAL).

Level 3.4 Connect Controller. This module is the actual connection of the system components and is performed in accordance with level 3.1.

Level 4.2 - Standard Design Techniques. This module contains no code. It is accomplished during Level 4.1 and is included here only for completeness.

Level 4.3 - Analyze Performance. This module contains no code. It is accomplished during Level 4.1 and is included here only for completeness.

Level 4.4 - Store Coefficients. This module stores the design parameters for the digital controller to be implemented.

The final controller design must be placed in the ICECAP variable HTF (29). When the design process is complete, the user leaves ICECAP with the 'STOP' command (29). This saves all of the design session's variables and parameters (including the controller design in HTF) in a data file called MEMORY.DAT within the VMS file system. This file will later be read by a DICES module in order to

determine the order of the controller and the poles and zeros. This data will be used by DICES to perform the remainder of the implementation of the digital controller.

Following this storage operation, ICECAP will return the user to the DICES main menu, with no required user input. From here the user may select a DICES option to continue on to the implementation step.

Level 5.1 - Retrieve Factors. This module retrieves the controller design parameters that have been stored in the data file MEMORY.DAT. The coefficients actually obtained are the poles, zeros, and a constant term which is the quotient of the numerator and denominator constants which have been factored out of each expression. The pole-zero data are read into a complex array and the constant term into a real variable. These data are then available to the DICES program for further manipulation.

ICECAP writes all variables into the sequential data file MEMORY.DAT. Up to this point in DICES, no information is available internally to DICES about the nature of the controller design. Therefore, since the order of the controller numerator and denominator, as well as the poles, zeros and constant term are embedded within this file, it must be read sequentially in order to extract them. To properly read the poles and zeros from MEMORY.DAT, the

order of the numerator and denominator must be known apriori. Unfortunately, ICECAP writes the order of the numerator and denominator after the poles and zeros, so the file must be read once through (past the unknown number of poles and zeros) to obtain the order. Then the file is REWOUND and read again to read in the proper number of poles and zeros.

MEMORY.DAT File Organization. The MEMORY.DAT file organization was investigated to determine the position of the required data items within the file. Since no explicit information existed to obtain the sequence of writes to the file (except possibly searching through many subroutines and trying to figure it out), it was decided to enter test cases into ICECAP and examine the MEMORY.DAT after each case until the position and format of the poles, zeros, polynomial coefficients, and constant term were determined. The following was the outcome of this exercise:

MEMORY.DAT consists of 721 lines of data, most of which contain five data items. Several other lines contain two, four, or six data items. The following lines all contain five data items per line unless otherwise noted. Sequencing through the data file, the first data of interest is a block starting at line 330 which contains the polynomial coefficients of the numerator of HTF. The position of a data item will be represented as [line #(data item #)]. If all the data items in a line are referenced, an 'x' will be used within the parentheses.

Example :

The position of the last coefficient for the numerator polynomial of HTF is written as [340(1)]. The entire line is written as [340(x)].

ICECAP allows HTF to be of order 50, so there are 51 coefficients in this block (many of which are usually zero). The block consists of lines [330(x)] to [340(1)].

The next block is from [340(2)] to [350(2)]. This block consists of the denominator polynomial of HTF which is also of maximum order 50.

The next block from [350(3)] to [360(2)] contains 50 data items and consists of the real part of the zeros of HTF.

The next block from [360(3)] to [370(2)] contains 50 data items and consists of the imaginary part of the zeros of HTF.

The next block from [370(3)] to [380(2)] contains 50 data items and consists of the real part of the poles of HTF.

The next block from [380(3)] to [390(2)] contains 50 data items and consists of the imaginary part of the poles of HTF.

Data item [390(3)] contains the order of the numerator of HTF.

Data item [390(4)] contains the order of the

denominator of HTF.

Data item [390(5)] contains the constant term K that is factored out of the numerator and denominator in order to put the controller transfer function in the form

$$G_{hc} = \frac{K(z + z_1) \dots (z + z_h) \dots (z + z_m)}{(z + p_1) \dots (z + p_i) \dots (z + p_n)} \quad (2)$$

where K is the loop sensitivity, z_h is a real or complex zero, and p_i is a real or complex pole (5:220).

HK is not updated during changes after using the RECOVER option, so $HK = HNK/HDK$ is used (line 391).

Data item [615(4)] contains the sample period used to perform the S-plane to Z-plane transformations.

The following table summarizes the positions of the required data:

[330]	a_m	a_{m-1}	a_{m-2}	a_{m-3}	a_{m-4}
.
[340]	a_0	b_n	b_{n-1}	b_{n-2}	b_{n-3}
.
[350]	b_1	b_0	Rez_m	Rez_{m-1}	Rez_{m-2}
.
[360]	Rez_2	Rez_1	Imz_m	Imz_{m-1}	Imz_{m-2}
.
[370]	Imz_2	Imz_1	Rep_n	Rep_{n-1}	Rep_{n-2}
.
[380]	Rep_2	Rep_1	Imp_n	Imp_{n-1}	Imp_{n-2}
.
[390]	Imp_2	Imp_1	m	n	K
[391]	HNK	HDK	x	x	x
.
[615]	x	x	x	TSAMP	x

Table 11. MEMORY.DAT Organization

where

- a_h = coefficient of numerator polynomial term of Z^h
- b_i = coefficient of denominator polynomial term of Z^i
- Rez_h = real part of zero h
- Imz_h = imaginary part of zero h
- Rep_i = real part of pole i
- Imp_i = imaginary part of pole i
- m = order of numerator
- n = order of denominator
- K = loop sensitivity coefficient
- TSAMP = Sample period

Level 5.5 - Re-analyze. This module retrieves the stored quantized coefficients and places them into ICECAP's data file called MEMORY.DAT. When ICECAP is re-entered and the 'RECOVER' option is selected (29), the factors used for the ICECAP session are the quantized values. This allows the user to ascertain the effects of quantization on the performance of the controller.

The current version of ICECAP does not re-factor the polynomial coefficients when a RECOVER is performed. Since DICES stores the coefficients of first and second-order sections, they must be multiplied out to obtain the new full order numerator and denominator polynomials. This new polynomial must also be factored by DICES and the roots stored in the appropriate MEMORY.DAT locations as well.

DICES has the numerator and denominator first and second-order polynomials stored. These can be factored easily using the familiar quadratic equation formula and stored as mentioned above.

Level 6.1 Test Requirements. This module must present to the user the various options available for testing the closed-loop performance of the control system.

The first menu presented requests the type of test to be performed on the system. Once the test is selected, the program asks for any relevant data. The tests currently available are the step response and closed-loop frequency response (magnitude and phase). Additional tests can be

added at a later date.

User-supplied data for the step response test consist of

- 1) Amplitude of step (volts)
- 2) Approximate settling time of system

User-supplied data for the frequency response test consist of

- 1) Approximate closed-loop bandwidth

Step Response. This test will measure the step response of the closed-loop system. A display is presented to allow the input of values associated with the step input that will be input to the SUT. These values are used to program the signal generator and oscilloscope function of the B/K 2032 via the IEEE-488 interface.

Frequency Response (Magnitude). This test will measure the magnitude versus frequency response of the closed-loop system. The user inputs the approximate bandwidth of the system to allow proper configuration of the B/K System Analyzer via the IEEE-488 interface bus. The entire frequency response test is performed by the B/K 2032. The approximate bandwidth of the closed-loop system is used to determine the frequency span on the B/K 2032.

Frequency Response (Phase). This test will measure the phase response versus frequency of the closed-loop system. The approximate bandwidth of the closed-loop system is used to configure the B/K System Analyzer.

Level 6.5 Collect Test Results. This module will collect the results from the various tests performed by the system and display the results as printed copies, graphics displays or text displayed on the computer terminal.

The frequency response tests will use the B/K 2032 System Analyzer to perform the tests. This instrument has the capability of generating white-noise which is used to obtain frequency and phase data from the SUT. The display of the magnitude and phase response of the System-Under-Test (SUT) will be displayed on the graphics display of the B/K 2032.

The step response of the SUT will be displayed on the B/K 2032 with rise-time and settling time measurements made using the delta-time capabilities of the B/K 2032.

The impulse response of the SUT will be measured and displayed by the B/K 2032 System Analyzer. This instrument has a narrow-width pulse generator which is used to approximate an impulse into the SUT and then display the response of the SUT.

Detailed Filter Implementation Module Design

Level 5.2 - Pick Filter Structure. This module selects the filter structure to be used to implement the digital compensator (4,6,8).

There are many basic structures that can be used to implement a digital filter. There are 1D, 2D, 3D, 4D, 1X, 2X structures, ladder and continued fraction structures. The 1D, 2D, 3D, and 4D are known as 'Direct' structures because the coefficients show up explicitly in the implementation of the filter. The 1X and 2X are referred to as cross-coupled structures and can be used to realize complex pairs of poles and zeros.

The user should be presented with the options available and select the one desired. If the user is not familiar with the various structures, one shall be selected as the default structure type.

Default Structure. The controller is to be implemented as a cascade of first and second order sections, therefore each section must be realized by one of the above structures. The 1D, 2D, 1X, 2X, ladder, and continued fraction structures are canonical structures because they contain the minimum number of delay elements, while the 3D and 4D structures are non-canonical (13:357).

Each of the structures have some time and/or space advantage over the others. For example, the canonical forms use the minimum number of delay elements possible, while

the non-canonical structures use more than the minimum number.

In applications such as dedicated Very-Large-Scale-Integration (VLSI) DSP integrated circuits, the number of data lines, delay elements, summing junctions, multipliers, length of connections are very important. These all translate into 'real estate' that must be used to implement the additional devices. It may be desirable to use a canonical form when faced with the above situation.

When implementing a digital filter using a general-purpose microprocessor or DSP processor, many of these important factors mentioned above become much less important. For example, in a microprocessor, the number of summing junctions is not critical because the microprocessor need only add more numbers to a register or storage location. Of course, this may add time, so that a trade-off analysis may be necessary.

Selection of Default Structure. The most straightforward structure to implement is the 3D structure, since it is a direct implementation of the difference equation.

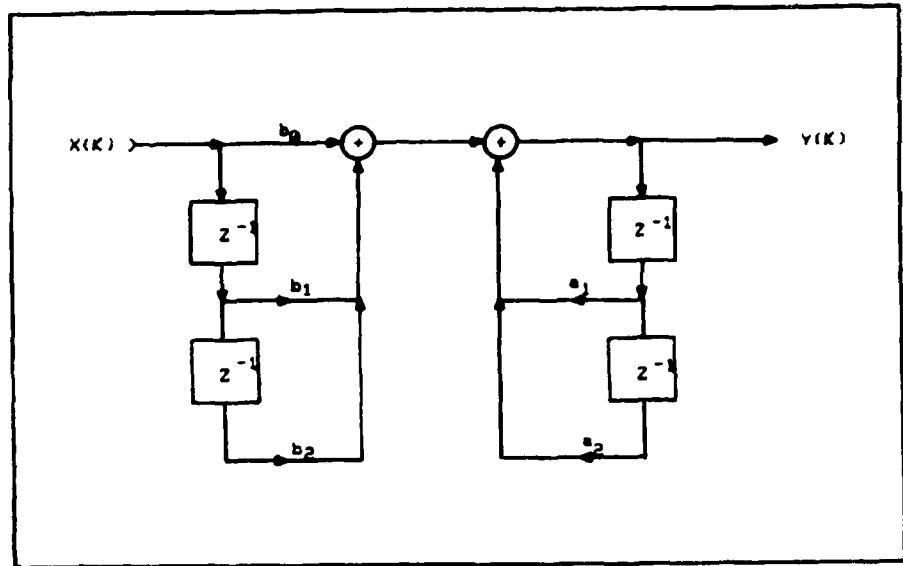


Figure 13. 3D Filter Structure

The transfer function is

$$\frac{Y(Z)}{X(Z)} = \frac{b_0 + b_1 Z^{-1} + b_2 Z^{-2}}{1 - a_1 Z^{-1} - a_2 Z^{-2}} \quad (3)$$

Taking the inverse Z-transform and solving for Y(k)
yields

$$Y(k) = b_0 X(k) + b_1 X(k-1) + b_2 X(k-2) \\ + a_1 Y(k-1) + a_2 Y(k-2) \quad (4)$$

As can be seen, this equation for $Y(k)$ is straightforward to implement. Because the 3D structure is not canonical, there are extra delay elements. This poses no problem because of the TMS32010's extremely high speed (most instructions execute in 200 nanoseconds) and over 4 kilowords of program memory (3).

A typical second-order section with 16 bit coefficients may take 12 to 14 instructions to compute that section's output, $Y(k)$. This is approximately 2.4 - 2.8 microseconds per section (exclusive of non-recurring initialization routines).

For most physical systems that DICES will be used for, the fastest time constants of the system are probably on the order of 10^{-2} seconds. Assuming a delay of 1/20th the fastest time constant is acceptable (5 %), this yields

$$\tau_c = 500 \text{ } \mu\text{secs} \quad (5)$$

where τ_c is the computational delay of the filter.

The number of sections that can be cascaded before the delay is excessive is

$$1 = \frac{500 \text{ } \mu\text{secs}}{2.5 \text{ } \mu\text{secs/section}} \quad (6)$$

$$1 = 200 \text{ sections} \quad (7)$$

It can be seen that there are no real detrimental effects

due to the 3D structure.

The 3D structure will be selected as the default structure because of its ease of implementation, straightforward debugging, and speed. This is certainly not the optimal choice, but it will provide a basic structure which will allow DICES to 'get off the ground'. Additional filter structures may be added later if desired.

Maximum Filter Order. An effort was made to keep the design free of anything that would prevent arbitrary order controllers. A choice was needed, however, in order to establish a baseline system from which to deviate from.

It has been proposed (13) that for a 16-bit filter implementation, coefficient quantization does not cause significant problems if the order of the filter is less than 10. In an effort to establish a baseline, a maximum order of eight was chosen for the basic DICES. There are no known fundamental limitations to this limit within DICES, other than speed and memory size. This limitation can be changed if and when desired.

Level 5.3 - Pole/Zero Pairing. When implementing a filter with first and second-order sections, an obvious question arises - how to select the poles and zeros that go into each section?

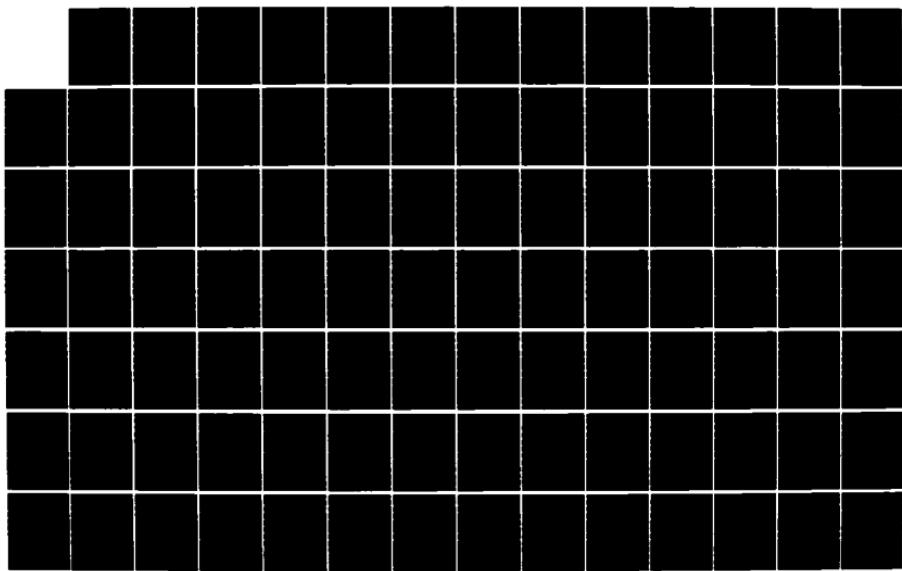
An optimal pairing could be performed by minimizing a

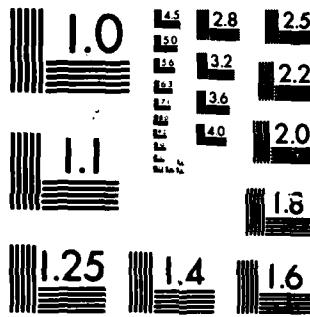
AD-A163 966 DEVELOPMENT OF A DIGITAL INTERACTIVE CONTROLLER
EVALUATION SYSTEM (DICES)(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
UNCLASSIFIED S B ECKERT DEC 85 AFIT/GE/ENG/85D-13

2/3

F/G 9/5

NL





MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS 1963-A

very lengthy contour integral proposed by Phillips and Nagle (13:481). This approach consists of $[n/2]^2$ (note 1) evaluations of the integral. For the case of $n = 8$, this yields 576 evaluations that must be performed! It has been suggested (13:481) that a simplified algorithm which utilizes graphical techniques gives similar results to the 'optimal' solution. This algorithm will be used here with slight modifications to enable implementation within a computer program, but the basic algorithm remains unchanged.

If future work on DICES is undertaken, this pole/zero pairing algorithm may be modified, but it is doubtful that much will be gained by implementing the 'optimal' solution.

The algorithm to be used is as follows:

1. Form Array. Poles and zeros are formed into arrays for easy access.

2. Pair Real Poles. Find real pole nearest $Z = +1$ point. Pair it with the real pole farthest from the $Z = +1$ point. Continue until all the real poles have been paired. All complex poles remain together.

3. Pair Poles and Zeros. Find pole nearest $Z = +1$. Match it with zero nearest its location. If the zero is real, match the other pole of the pole-pair in the same manner. If zero is complex, use conjugate. Repeat step 3 until all poles and zeros have been matched. Use a '1' in numerator when zeros exhausted.

note 1: $[n/2] =$ smallest integer greater than $n/2$.

An example of this process follows:

Consider the following fourth-order filter of figure 14.

$$G(z) = \frac{(z - .9156 \pm j.4021)(z + .6112)z}{(z - .8818 \pm j.3548)(z - .2124)(z - .3412)} \quad (8)$$

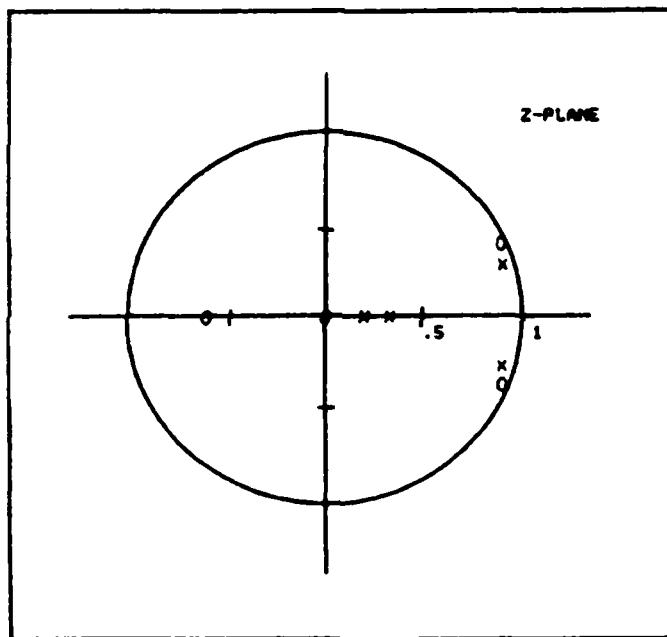


Figure 14. Example Filter

Applying the algorithm to this filter yields the pairing as shown.

Filter Section 1 :

$$\frac{(Z - .9156 \pm j.4021)}{(Z - .8818 \pm j.3548)} \quad (9)$$

Filter Section 2 :

$$\frac{(Z + .6112)Z}{(Z - .2124)(Z - .3412)} \quad (10)$$

These results are the same as that produced by Jackson (24) using a different 'near optimal' algorithm.

The algorithm is now discussed in more detail using flowcharts to show the major functions required to perform the pairing process.

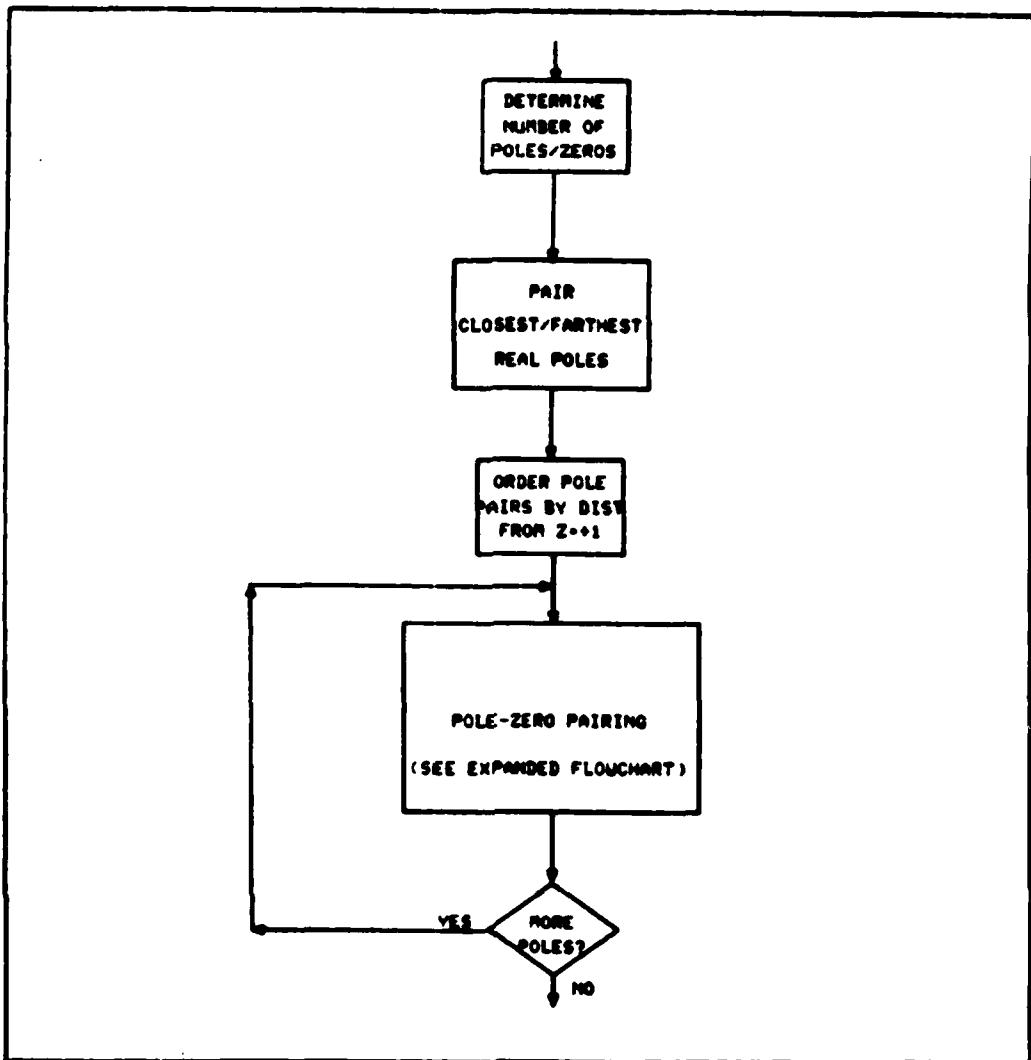


Figure 15. Pairing Algorithm

From the flowchart, it can be seen that the main functions to be performed are:

1. Determine number of zeros and poles
2. Pair closest and farthest poles
3. Order all poles by distance to $Z = +1$

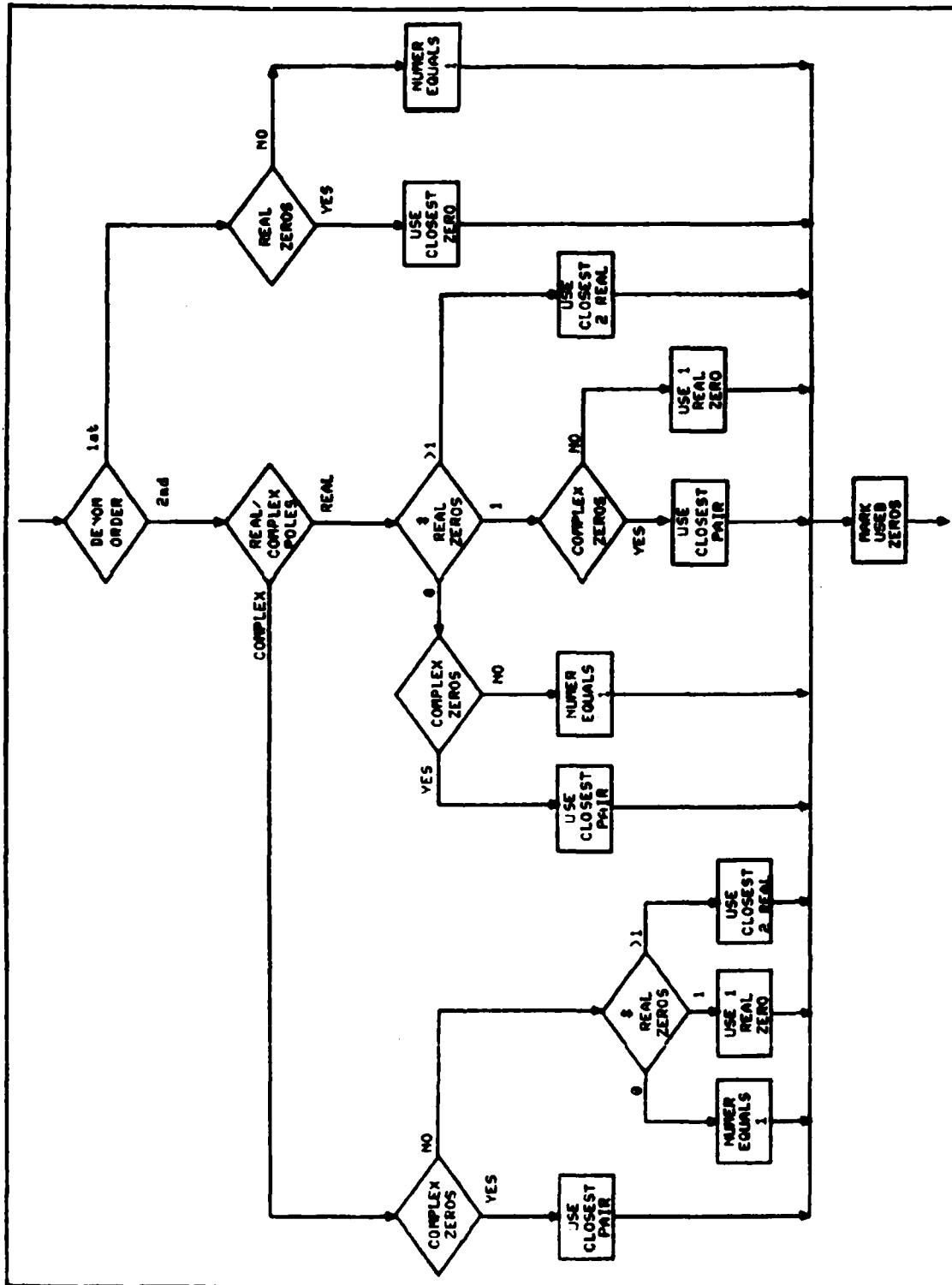


Figure 16. Pairing Algorithm

4. Pair zeros with poles
 - a. Determine if there are any complex zeros
 - b. Find closest pair of complex zeros to poles
 - c. Find number of unused real zeros
 - d. Find closest 2 real poles to complex zeros
 - e. Find closest real zero to pole

These major functions will be implemented as subroutines that will be called from major modules to perform the pole-zero pairing.

Level 5.4 - Quantize Coefficients. This module quantizes the digital controller coefficients which were determined by the design process using ICECAP.

From other previous processes, the controller has been partitioned into first and second-order sections. Each of these sections will be implemented as a filter using the structure determined elsewhere (1D, 2D, etc.). The roots of the sections are contained within a complex array called CORDPZ. The format of the array is as follows:

<code>zero(Re)</code>	<code>zero(Im)</code>
<code>zero(Re)</code>	<code>zero(Im)</code>
<code>pole(Re)</code>	<code>pole(Im)</code>
<code>pole(Re)</code>	<code>pole(Im)</code>
.	.
.	.
<code>zero(Re)</code>	<code>zero(Im)</code>
<code>zero(Re)</code>	<code>zero(Im)</code>
<code>pole(Re)</code>	<code>pole(Im)</code>
<code>pole(Re)</code>	<code>pole(Im)</code>

Table 12. Array of Controller Roots

These sections are ordered with the first section to be computed at the top of the array, and the last section at the end of the array.

Quantization. The effect of coefficient quantization is a 'migration' of the poles and zeros of the transfer function. After quantization, as the order of the controller increases, the movements of the roots away from the desired roots becomes more and more pronounced. It is well known (14:441) that the roots of a polynomial become more sensitive to parameter changes as the order of the polynomial increases. This effect produces a 'new' controller design - which may or may not meet

specifications or even be stable. This effect is the primary reason for implementing high-order (>2) controllers as cascade sections of first and second order.

Quantizing the coefficients of a digital filters effectively restricts the poles and zeros of the filter to lie on a finite number of discrete points in the Z-plane (14:441). For example, consider the following second-order filter:

$$G(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \quad (11)$$

In general, the roots of a second-order denominator polynomial are given as

$$p_1 = (\sigma_1 + j\omega) \quad (12)$$

$$p_2 = (\sigma_2 - j\omega) \quad (13)$$

which can be rewritten in polar form as

$$p_1 = r_1 e^{j\psi} \quad (14)$$

$$p_2 = r_2 e^{j\psi} \quad (15)$$

$$\text{where } r_1 = [\sigma_1^2 + \omega^2]^{1/2}$$

$$r_2 = [\sigma_2^2 + \omega^2]^{1/2}$$

$$\psi = \tan^{-1}[\omega/\sigma] \quad \sigma_1 = \sigma_2 \text{ if complex roots}$$

The equation can be written as (assume $b_0 = 1$)

$$(Z + r_1 e^{jv})(Z + r_2 e^{-jv}) = \\ Z^2 + r_1 e^{jv}Z + r_2 e^{-jv}Z + r_1 r_2 \quad (16)$$

($r_1 = r_2$ if complex)
($v=0$ if real)

$$= Z^2 + (r_1 + r_2) \cos(v)Z + r_1 r_2 \quad (17)$$

$$= Z^2 + a_1 Z + a_2 \quad (18)$$

where $a_1 = -(r_1 + r_2) \cos(v)$

$$a_2 = -r_1 r_2$$

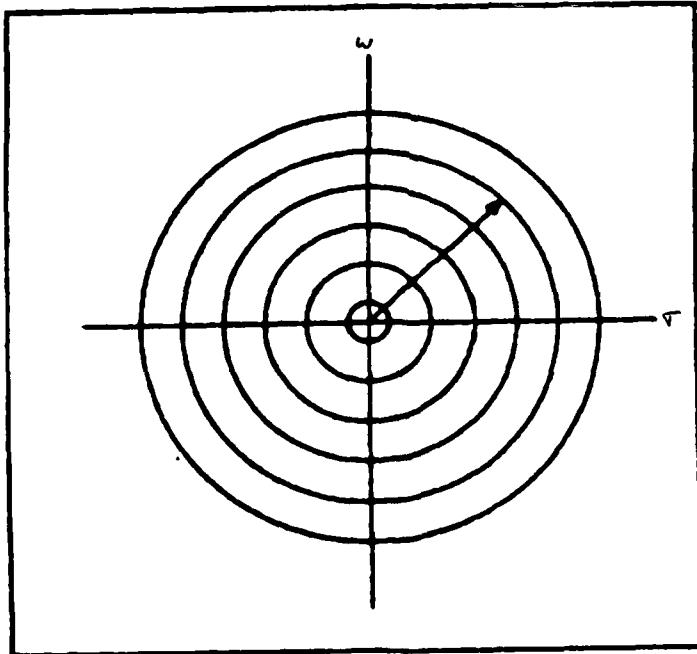


Figure 17. Circles of radius $[a_2]^{1/2}$

If a_1 is quantized, it becomes a_1^q and a_2 becomes a_2^q , where a_i^q is the q-bit quantized version of a_i . Since $a_2 = r_1 r_2$, a_2 is restricted to a finite number of circles of radius $r = [a_2^q]^{1/2}$.

Coefficient a_1 is furthermore restricted to $a_1^q = -(r_1 + r_2) \cos(\nu)$ which is a finite number of vertical lines.

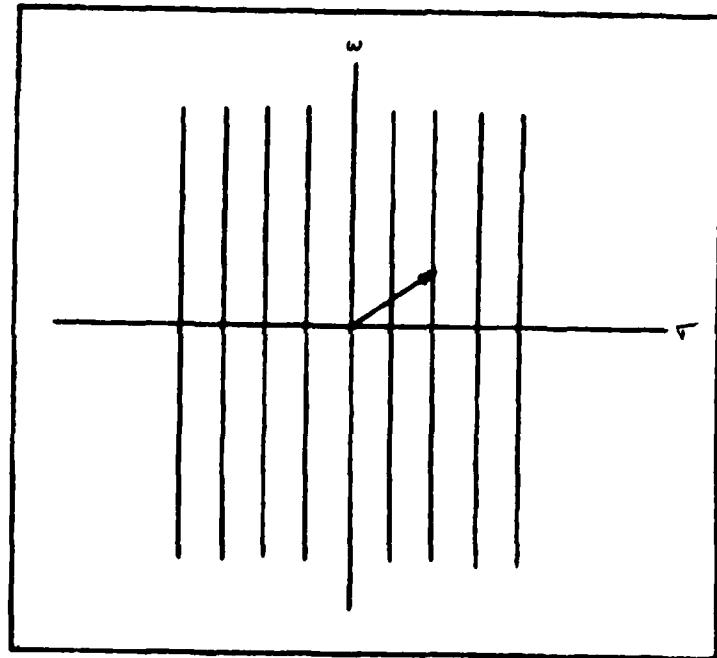


Figure 18. Vertical lines

The only possible pole-zero locations therefore lie at the intersection of these two figures.

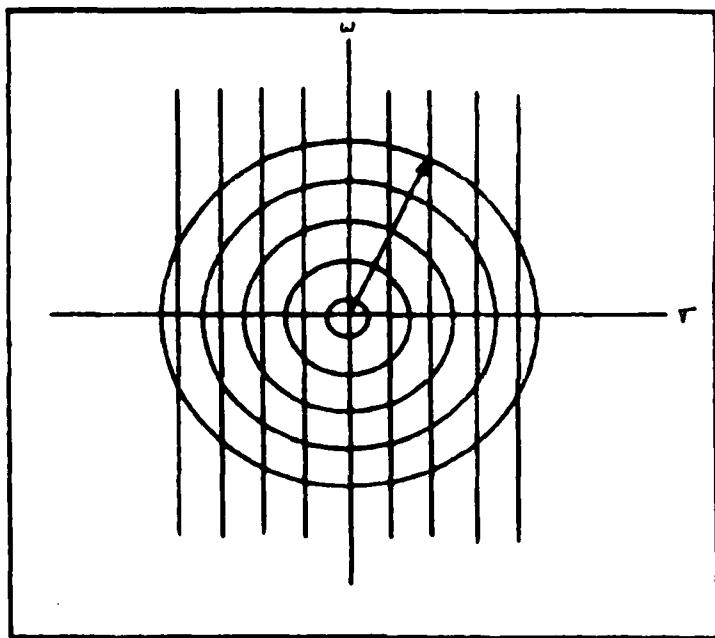


Figure 19. Intersection of figures 17 and 18

Since the TMS32010 uses 16-bit signed two's-complement arithmetic, the filter coefficients will be quantized to 16 bits. The format used in the TMS32010 is what is known as Q15 (4). Q15 notation has a single sign bit in the Most-Significant-Bit (MSB) position, followed by 15 two's-complement magnitude bits.

MSB LSB
S.XXXXXXXXXXXXXXXX

This notation allows a number representation of between +.999969 and -1.0 in Q15 notation, with increments (resolution) of .000031. However, in the TMS32010, only integer arithmetic can be performed. These numbers have a range of +32767 to -32768.

The relatively high precision coefficients from ICECAP, c_i , must be multiplied by 2^{15} (32768) to move the binary point after the sign bit 15 places to the right. The coefficient is then truncated or rounded to obtain an integer between the limits given previously.

$$[c_i] = c_i * 2^{15} \quad (19)$$

$$[c_i]^T = \text{Truncate } [c_i * 2^{15}] \quad (20)$$

$$[c_i]^R = \text{Round } [c_i * 2^{15}] \quad (21)$$

where $[c_i]$ is the coefficient multiplied by 2^{15} and $[c_i]^T$ or $[c_i]^R$ is the truncated or rounded integer version of the coefficient c_i .

Example: Let

$$a_2 = .756761$$

$$[a_2] = .756761 * 2^{15} = 24797.544$$

$$[a_2]^T = 24797 \text{ (Truncation)}$$

$$[a_2]^R = 24798 \text{ (Rounding)}$$

Now, to convert back to its Q15 equivalent, the number must be divided by 2^{15} . Because of the loss of some of the precision of the number due to truncation or rounding, the 'new' number will not equal the 'old' number.

For example

$$a_2'{}^T = 24797/2^{15} = .7567443 \quad (22)$$

$$a_2'{}^R = 24798/2^{15} = .7567749 \quad (23)$$

(where the prime indicates Q15 notation)

produces a difference (from the original coefficient) of -.0000167 for the truncated coefficient and +.0000139 for the rounded coefficient.

For 16-bit implementations, coefficient quantization will generally not be a problem on the overall performance of the compensator, provided the order of the filter remains less than 10 (8:32).

Before the coefficients can be quantized, they must be formed into a polynomial from the poles and zeros of each section. The array of filter poles and zeros for each section must be multiplied out to produce a first or second-order polynomial in Z, which will be implemented using a particular structure (1D, 2D, etc.). Quantization can then be performed on the polynomial coefficients of the resulting quantized first or second-order sections. When

the zeros and poles of the first section are multiplied out, the following results:

Poles

$$(Z + r_1 e^{jv})(Z + r_2 e^{-jv}) = Z^2 + r_1 e^{jv}Z + r_2 e^{-jv}Z + r_1 r_2 \quad (24)$$

($r_1 = r_2$ if complex)
($v=0$ if real)

$$= Z^2 + (r_1 + r_2) \cos(v)Z + r_1 r_2 \quad (25)$$

$$= Z^2 - a_1 Z - a_2 \quad (26)$$

and multiplying by Z^{-2} yields

$$= 1 - a_1 Z^{-1} - a_2 Z^{-2} \quad (27)$$

where

$$v = \tan^{-1}[\omega/\sigma] \quad \sigma_1 = \sigma_2 \text{ if complex roots}$$

$$a_1 = -(r_1 + r_2) \cos(v)$$

$$a_2 = -r_1 r_2$$

Zeros

$$(Z + r_3 e^{jw})(Z + r_4 e^{-jw}) = Z^2 + r_3 e^{jw}Z + r_4 e^{+jw}Z + r_3 r_4 \quad (28)$$

($r_3 = r_4$ if complex)
($w=0$ if real)

$$= Z^2 + (r_3 + r_4) \cos(w) + r_3 r_4 \quad (29)$$

$$= Z^2 + b_1 Z + b_2 \quad (30)$$

where

$$w = \tan^{-1}[\omega/\sigma] \quad \sigma_1 = \sigma_2 \text{ if complex roots}$$

$$b_1 = (r_3 + r_4) \cos(w)$$

$$b_2 = r_3 r_4$$

And finally, multiplying by the numerator loop sensitivity, which is b_0 , yields

$$G(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \quad (31)$$

Each of the above coefficients is then quantized using the previously discussed method, which produces

$$G(z) = \frac{b_0' + b_1' z^{-1} + b_2' z^{-2}}{1 - a_1' z^{-1} - a_2' z^{-2}} \quad (32)$$

where the prime ('') on the coefficients indicates either a rounded or truncated quantized coefficient.

Each of the above quantized coefficients has an integer counterpart which is represented by $[c_i]^T$ or $[c_i]^R$. From this point forward, a quantized coefficient will be

represented by c_i' , which is Q15 notation, but the coefficient is always represented as an integer within the TMS32010.

After quantization, the coefficients are stored in a file for later use in generating the TMS32010 code to implement the filter.

Level 5.6 - Scaling and Ordering. The coefficients must be scaled and the second-order sections ordered in order to minimize the probability of overflow and to minimize error due to coefficient quantization.

The order of the sections will be as determined by the pairing algorithm. While this is not an optimal ordering, it is an easily implemented method. If this thesis investigation were to be continued, an optimal ordering algorithm would be a possible follow-on topic.

Scaling the filter coefficients is a means to prevent the digital filter from overflowing internally while computing the next filter output, $Y(k)$. The current version of DICES performs no scaling other than that which distributes the gain equally over the number of sections used.

Level 5.7 - Sample Period Selection. This module provides the user with the option to either select the sample period desired (within the limits of the AIB) or have DICES extract the sample period that was used during

the design process using ICECAP.

As discussed previously, TSAMP is obtained from line 615(4) of the ICECAP data file MEMORY.DAT. This value is read into a variable called TSAMP and should be checked for compatibility with the AIB board. After quantization, this variable is stored along with the quantized coefficients.

Hardware Limitations. The AIB board has a programmable divide-by-N rate multiplier that can be set to count down to zero from a preset number and upon completion, signal the TMS32010 by pulling a pollable pin low (BIO) (3). This pin can be monitored to produce very precise sample rates in order to read an input sample and calculate the next output value in a very periodic manner. Since the clock used to drive the TMS32010 is crystal controlled, the time intervals are very stable.

The circuit used on the AIB board is loaded with a 16-bit number which represents a multiplier to be used in the clock circuit. The input clock frequency used for the count-down is 5.0 Mhz, which is a period of .2 microseconds. The users' manual for the AIB board gives the following equation to determine the number 'N' to load to produce a particular interval.

$$F_{out} = \frac{(N) * F_{in}}{2^{18}} \quad (33)$$

$$= N * 19.073486 \quad (34)$$

The frequency must be within the limits

$$19.073486 \text{ Hz} < F_{\text{out}} < 40,000 \text{ Hz} \quad (35)$$

The lower limit is set by the minimum value for N of 1. The upper limit is restricted because of the AIB A/D converter speed of 25 microseconds. The sample period range is therefore

$$25 \text{ microsecs} < T_d < 52.42 \text{ millisecs.} \quad (36)$$

and the range of integer N is

$$1 < N < 2097 \quad (37)$$

Equation (34) can be rewritten as

$$N = \frac{F_{\text{out}}}{19.073486} \quad (38)$$

$$N = \frac{1}{T_d * 19.073486} \quad (39)$$

where

$$F_{\text{in}} = \text{CLKOUT (5.0 Mhz)}$$

N = constant value in clock register
($0 < N < 2097$) . N should be a power of two if a stable sampling rate is desired.

F_{out} = output frequency of the pulse at the BIO pin.

T_d = desired sample period which must be between limits specified. $T_d = 1/F_{out}$

Since the counter is limited to 16 bits in conjunction with the clock rate used, the range of sample periods available is 25 microseconds to 52.42 milliseconds as shown above. This is not slow enough for typical control system applications that involve somewhat slow system time-constants. A range of 100 Hz (10 milliseconds) to about .1 Hz (.1 seconds) is more reasonable for most physical control applications and will be used here.

In order to accomodate this new range of sample periods without hardware modifications, a software timing loop will be incorporated into the filter program which will effectively extend the range to the above requirements. This will be accomplished by acting upon only every 400th occurrence of the BIO pin signalling the end of a period. This will effectively increase the sample-period span from 25 milliseconds to 20.97 seconds.

When the variable TSAMP is read from MEMORY.DAT, it must be checked to see if it falls within the bounds above. If not, the user must be notified and given the option to escape and re-do the design. If the sample period is acceptable, it must be divided by 400, since it is the actual sample period desired. This produces a value within

the range of the AIB and the multiplication by 400 is done in the filter software by accepting only every 400th countdown.

Example:

If a TSAMP of 1 second was used for design and DICES reads 1 second as the TSAMP value, this value would be outside the range of the AIB board. It is divided by 400 to produce .0025 seconds (2.5 milliseconds). This value produces an 'N' of

$$N = \frac{1}{T_d * 19.073486} \quad (40)$$

$$= \frac{1}{.0025 * 19.073486} \quad (41)$$

$$= 20.97 \quad (42)$$

Since N must be an integer, N = 21 is used. This produces a sample period of T = 2.4966 milliseconds and equates to a final sample period of .99864 seconds.

The following table was derived by picking nominal values of T_d , dividing by 400, and computing 'N' using equation (39). Taking the integer portion and using equation (33) produces the actual F_{out} . The reciprocal of

this F_{out} is taken and multiplied by 400 to obtain the actual time, T_s . The table shows the degradation in accuracy as the sample-period increases. This is because N must be an integer, and loading the lowest value of '1' into the AIB produces the longest period - 52.42 milliseconds (because the counter is a divide-by-N counter - i.e., it divides the input signal by N). Multiplying this by 400 yields 20.97 seconds. If N = 2 is loaded, an interval of 26.21 milliseconds results and 400 times this produces 10.48 seconds.

NOMINAL TIME(T_d)	ACTUAL TIME(T_s)	% ERROR
.1	.100055	0.05 %
.3	.299593	-0.14 %
.5	.499322	-0.14 %
.7	.699051	-0.14 %
.9	.903945	0.44 %
1.1	1.09227	-0.70 %
1.3	1.31072	0.82 %
1.5	1.49797	-0.14 %
1.7	1.69125	-0.51 %
1.9	1.87246	-1.45 %
2.3	2.27951	-0.89 %
2.7	2.75941	2.20 %
3.1	3.08405	-0.51 %
3.5	3.49525	-0.14 %
3.9	4.03298	3.41 %
4.3	4.36907	1.61 %
4.7	4.76626	1.41 %
5	5.24288	4.86 %
8	7.48983	-6.38 %
11	10.4858	-4.67 %
14	13.1072	-6.38 %
17	17.4763	2.80 %

Table 13. Sample-Period Errors

So, it can be seen that because of the requirement to load an integer, the accuracy of the desired sample period suffers at the slower rates. Below 10 seconds yields fair results with periods below 1 second yielding good results.

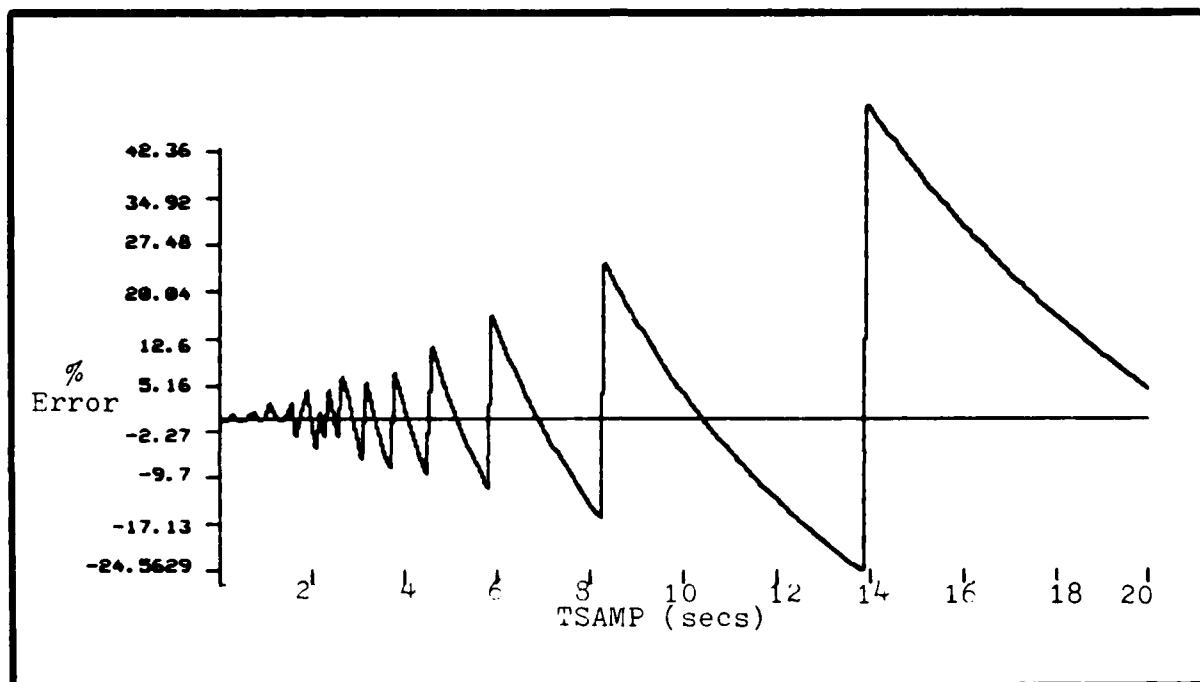


Figure 20. Sample-period errors

If better correlation is desired between the desired sample period and the actual sample period, the user can pick an integer for 'N' and determine what actual time it yields

using equation (39). After multiplying this time by 400, the user can then use the resultant time in the design phase and when converted by DICES during implementation, the correct value will result.

Level 5.8 - Generate Code. When the coefficients and sample period have been quantized, they must be inserted into the TMS32010 filter program. The filter program consists of a TMS32010 source file which is assembled by the TMS32010 Assembler after the coefficients have been inserted into the source code.

The filter program (called 3DFILT.THS) contains the filter coefficients and sample period constant in lines 168 to 188. These lines consist of 21 integers in the range of +32767 to -32768 along with the DATA statements in TMS32010 assembly language.

[168]	CB01 DATA XXXXX
[169]	CB11 DATA XXXXX
[170]	CB21 DATA XXXXX
[171]	CA11 DATA XXXXX
[172]	CA21 DATA XXXXX
[173]	CB02 DATA XXXXX
.	.
[186]	CA14 DATA XXXXX
[187]	CA24 DATA XXXXX
[188]	SMP DATA XXXXX

Table 14. Source File Structure

The source file is read in by DICES and written back out to a temporary file until line 168 is encountered. This line is then modified by retrieving the appropriate coefficient from the data file NEWCO.THS, which contains the quantized filter coefficients and sample period. This continues until all 21 integers have been loaded into the source file and written out to a temporary file. When the 21 data items have been modified, the remainder of the source file is transferred to the temporary file. The temporary file is then renamed to become the original 3DFILT.THS file again.

Assembly. When the data modifications are complete, the file 3DFILT.THS is a TMS32010 source program that produces the desired digital controller transfer function when assembled and executed on the TMS32010. DICES will return the user to the sub-menu for Filter Implementation. To assemble the source code, DICES returns the user to the VAX VMS operating system where the TMS32010 assembler is executed to assemble the file 3DFILT.THS. The DICES User's Manual contains complete details on this process. Once the object code has been generated, it is stored in a file called 3DFILT.MPO. This file now contains TMS32010 executable object code.

Detailed Plant Modelling Module Design

Level 3.2 Patch-up Equations. This process is the actual wiring of the analog computer system in order to implement the mathematical plant model. The user must perform this function using any available references, outside expertise, laboratory personnel, etc. This is sometimes a 'tricky' operation, as anyone who has had a laboratory in basic control systems or analog computer simulation knows. It is a very important part of the overall hybrid simulation because a faulty plant simulation will produce closed-loop operation that may be totally unpredictable. It may also erroneously cause the controller to be blamed when it is not at fault.

A typical analog computer simulation diagram for a simple transfer function, $G_p(s)$, is given on the following page.

$$G_p(s) = \frac{2}{s(s + 1)(s + 2)} \quad (43)$$

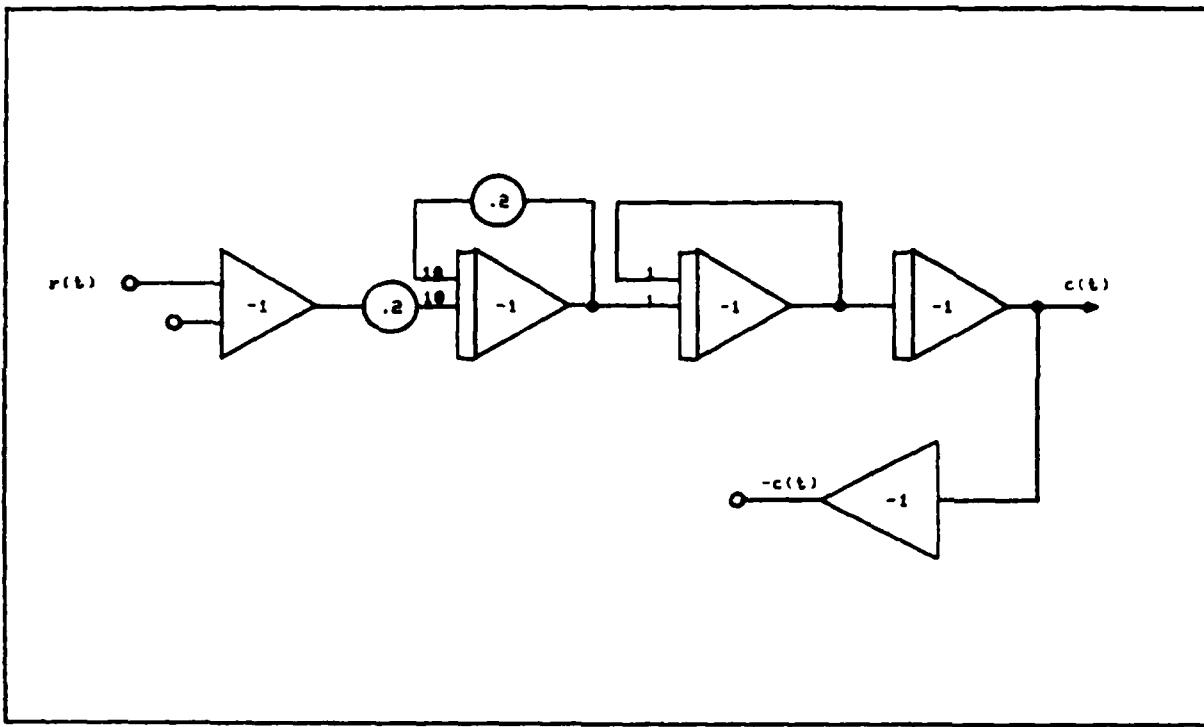


Figure 21. Analog Simulation Diagram

In this example, the simulation consists of:

- 3 integrators
- 2 amplifiers
- 2 potentiometers

The level of complexity is limited only by the number of integrators, amplifiers, and potentiometers available on

the analog computer. Additional analog computers can be used if desired.

Level 3.4 Connect System. This process is the physical connection of each of the pieces of hardware that make up the entire system (test equipment, plant, controller, etc.). Appendix A describes the system connection process in detail.

Using the block diagram below, each piece will be described as to its interfaces and means of connection to it's interfaces. First the connections common to both a cascade and feedback controller configuration are discussed, followed by the connections unique to each type of closed-loop control system.

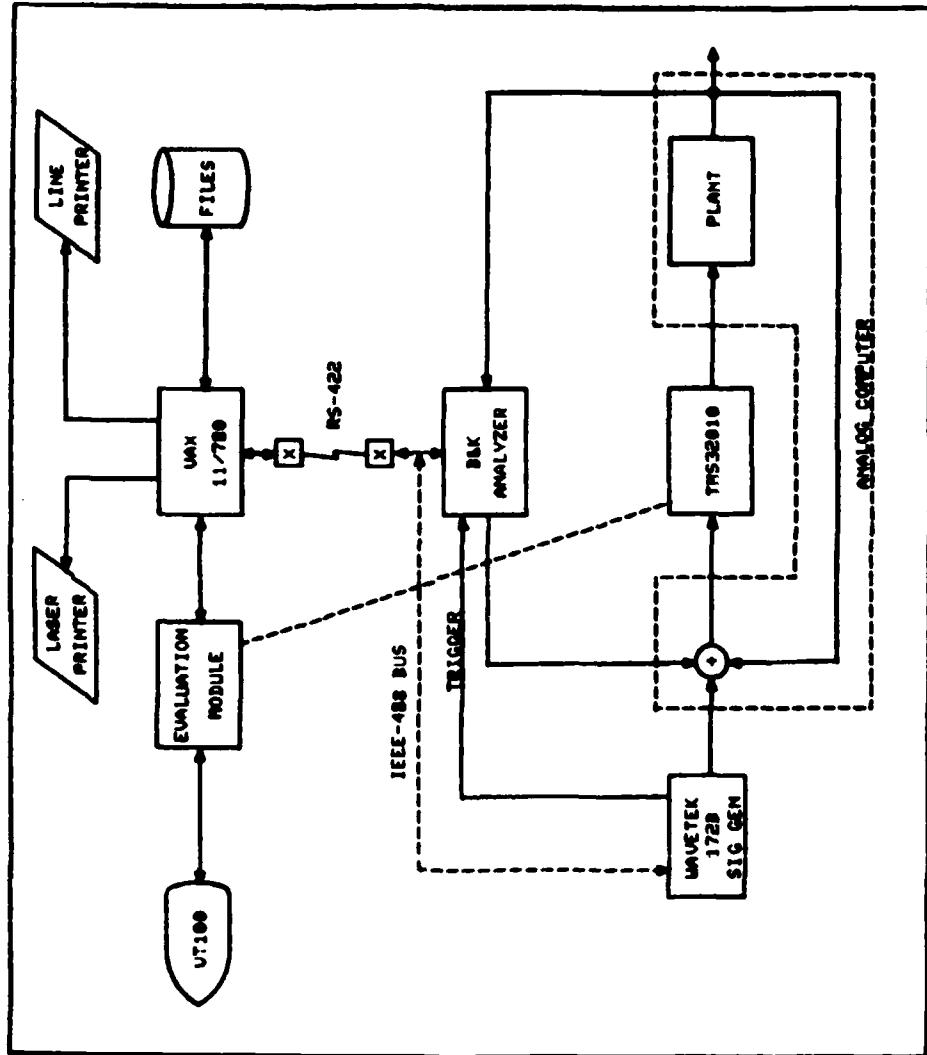


Figure 22. System Block Diagram

Evaluation Module. This item has two RS-232 connectors in the rear which are labelled, VAX and TRM.

The VAX connector is connected to the VAX via one of it's I/O ports. This VAX port must be configured by the system operator to 4800 baud and NOWRAP feature. This allows the downloading of TMS32010 object code to take place with no errors.

The TRM connector is attached to a VT100 video terminal (or similar). This terminal should also be configured to send and receive data at 4800 baud. The EVM firmware has been modified to send and receive data at 4800 baud. The EVM will sense the baud of the terminal at RESET, and if the terminal has not been set properly, the EVM will respond at the terminal rate. This may not cause any problems until the downloading of object code from the VAX occurs.

IEEE-488 Interfaces. The IEEE-488 cable from the National Instruments Bus Extender must be connected to the B/K 2032 Analyzer, and the Wavetek Signal Generator. Appendix A contains details on connecting these instruments to the IEEE-488 bus.

Signal Generator. The output of the Wavetek Signal Generator is connected to the input of the analog computer summing junction and to the B/K 2032 Trigger Input connector. These connections supply the reference signal

for step-response measurements.

B/K Analyzer. The analog computer output (which represents the closed-loop system response) is input to the analyzer. This signal is measured to obtain information about the response of the system under test.

Cascade Controller. This section discusses the configuration for a cascade controller.

EVM. The EVM is interfaced to the plant and summing junction of the closed-loop system via the two EVM front-panel BNC connectors. Note that the maximum input voltage to the EVM is ± 10 volts (2). Damage to the input sample-and-hold amplifier may result if this limit is exceeded.

The connector labelled INPUT is connected to the output of the summing junction on the analog computer. This output is the 'error signal' that drives the controller. An adaptor or special cable should be used to change the BNC from the EVM to the required connectors to interface with the analog computer.

The connector labelled OUTPUT is connected to the input of the plant simulation. This input is the first potentiometer, integrator, or amplifier that makes up the plant.

Plant. The input to the plant was connected when the EVM was connected to the analog computer. This provides the input signal to the plant. The plant output is connected to the B/K 2032.

Note that the unity feedback connection from the plant to the summing junction (subtracted from the inputs) is assumed to be a part of the analog computer simulation and is not addressed here.

Feedback Controller. This section discusses the configuration for a feedback controller.

EVM. The connector labelled INPUT is connected to the output of the plant simulation on the analog computer.

The connector labelled OUTPUT is connected to the input of an inverting amplifier on the analog computer. This signal, when inverted, becomes the negative feedback input to the summing junction.

Plant. The input to the plant is made internally on the analog computer. The input to the plant is the output of the summing junction on the analog computer.

Detailed System Configuration Module Design

Level 4.1 - Access CAD Package. This module must provide the user access to ICECAP. Any system data must be saved for later use when the user returns to the main DICES program.

This module should initiate execution of the ICECAP CAD design package with no input required from the user, other than the initial selection. Upon completion of the design phase, control should be passed back to DICES and the main menu again displayed.

Level 5.9 Load TMS32010 Code. This module deals with the downloading of the TMS32010 object code from the VAX to the EVM which, when executed, will implement the desired digital controller algorithm.

The EVM has a utility function called 'Load Program Memory (LPM)' which allows a remote system, such as the VAX, to download object code to the EVM's program memory. The format of the data is TMS9900 object code format, which is what the VAX TMS32010 assembler generates.

In order to load the object code into program memory, the file of object code must exist in the VMS file system. The VAX TMS32010 assembler generates a default object file name of 3DFILT.MPO.

Download Process. This section will describe in a general manner the downloading process. Full details are

given in the DICES User's Manual.

To start the downloading process, the user must exit DICES to the VMS operating system. The user then toggles the EVM out of the transparency mode into the EVM monitor program (remember that the EVM is in cascade with the user's terminal and the VAX). The command to Load Program Memory is given which causes the EVM to wait for data from the VAX. Control is returned to the VAX VMS system following this command. A VMS command to type the object file 3DFILT.MPO is given and the EVM begins loading the object code into its program memory. When complete, the user returns to the VMS system by again toggling the transparency mode. DICES is then started and the user continues with DICES.

Level 6.1 Determine Test Requirements. In addition to determining the test requirements from the user, this level also initializes the programmable test equipment in DICES.

Default Instrument Settings. The Wavetek 172B and B/K 2032 System Analyzer are programmed with settings that do not change from one test to another. The settings for the B/K can also be stored within the instrument in order to simplify the switching between test configurations.

The default parameters are now listed for the B/K 2032 and the Wavetek 172B.

B/K 2032 Default Parameters. The B/K 2032 is used for both step-response and frequency/phase response

testing. It must, therefore, have a different configuration for each of these two functions.

For the Step-Response test, the B/K must perform like an oscilloscope with a very wide range of sweep times. This is required to record even the slowest responding system response to a step input (limited to 512 seconds per sweep). To accomplish this function, there are several parameters that must be specified to the B/K (37). These settings are stored internally to the B/K 2032 as User Specified set-ups and can be recalled for later use, thereby saving time when the set-up is required later. These settings are described in the table below. A 'SYSTEM RESET 9' is performed prior to any parameter changes, since the reset configures the entire instrument to a known state. A description of the reset process can be found in the B/K 2032 operation manual (40:31). Only the deltas to the 'SYSTEM RESET 9' configuration are shown below.

<u>Function</u>	<u>Value</u>
Message Terminator	<LF>, 12 ₈
Trigger Mode	External
Trigger Slope	Positive
Display Function	Time
Channel	B
Channel Coupling	DC Direct

Table 15. Default Step-Response Settings

For frequency testing, the B/K must perform as a white-noise source and spectrum analyzer. These two devices working together are used to generate signals of all frequencies and measure the system's response (both magnitude and phase) to these signals.

The settings required for this mode of operation are standard settings already available in the B/K. There are no changes to this standard Frequency Response set-up (Setup #12). Only the Frequency Span must be changed for the particular system being tested. This function is performed in the Performance Evaluation module.

Wavetek 172B Default Parameters. The Wavetek 172B is used only for the step-response test. This instrument provides the step input to the system under test. The Wavetek 172B 'wakes up' from power-off in a state with the output disconnected (36). For step-response testing, the 172B must be configured to provide variable amplitude and duration pulses. These pulses must be compatible with the TR-48 analog computer system, so will generally be on the order of a few volts. The default settings that remain the same for all step response testing are shown in the table below. Note that it is not necessary to store these settings internally in the 172B because the 172B is not switched between modes for the various tests being performed.

<u>Function</u>	<u>Value</u>
Function	+ Pulse (0 v ref)
Mode	Triggered

Table 16. Step-Response Default Parameters

Detailed Performance Evaluation Module Design

This module consists of the software and hardware necessary to generate and send appropriate command strings to the test instrumentation and to initiate the test sequence.

Level 6.2 Generate Command String. In order to remotely program the test instruments used in DICES, the unique command strings for each piece of equipment must be generated. The test requirements that determine the command sequences are passed to this module and used to build the commands to be sent to each instrument. The current version of DICES permits two main tests to be performed - step response and frequency response (magnitude and phase).

General Programming Description. The B/K 2032 System Analyzer and Wavetek 172B are IEEE-488 programmable instruments. Each instruments' general programming methods are now discussed.

Wavetek 172B. Since the command format for the Wavetek 172B is described in detail in the Model 172B Programmable Signal Source Introduction Manual (36:17-32), only a summary is presented here.

- 1) A <LF> (linefeed) character is used as an end-of-string (EOS) character
- 2) Spaces <SP> are not allowed within data messages except as defined in the program code description

The general input data message format is as follows:

<program code¹><Numeric Data²>[program code³><Numeric Data>]...I⁴<LF⁵>

where the brackets indicate optional program codes and

- 1 Device dependent command
- 2 Numeric data required for function selected
- 3 Multiple device dependent commands are allowed
- 4 Execute command
- 5 End-of-String message

Program codes consist of a single upper-case character function identifier (i.e., prefix) and a parameter field. Suffixes are not used in 172B program codes. The one-letter

identifier defines the major instrument function being addressed. Depending on the identifier, the associated parameter field may contain zero, one, or multiple parameters. The general rules of program code format are:

- o The 172B sends and receives data messages in standard ASCII format.
- o The instrument responds only to upper-case characters.
- o Program codes are space sensitive

Numeric data consists of the characters 0 through 9, E, -, and decimal point (.).

B/K 2032 System Analyzer. Detailed programming instructions are contained in reference 37, so only a summary of the command structure is presented here. The general format of the command structure is similar to that of the Wavetek 172B except that manufacturer-unique terminology is used.

- 1) Semicolons, commas (,), or spaces <SP> may follow the two-letter function identifier
- 2) A <LF> (linefeed) character is used as an end-of-string (EOS) character
- 3) Spaces <SP> are not allowed within data messages except as defined in the program code description
- 4) Carriage returns <CR> are not required before <LF>

The general input data message format is as follows:

<program code¹><LF²>

where

- 1 Device dependent command
- 2 End-of-String message

Program codes consist of a two-letter function identifier (i.e., prefix) and a parameter field. Suffixes are not used in B/K 2032 program codes. The two-letter identifier is a mnemonic that defines the major instrument function being addressed. Depending on the identifier, the associated parameter field may contain zero, one, or multiple parameters. The general rules of program code format are:

- o The 2032 sends and receives data messages in standard ASCII format.
- o The instrument responds equally to upper and lower case characters.

Two primary program codes are used, these are: ASCII character strings and binary data. Numeric data may be integer, decimal, and exponential, all of which may be signed or unsigned.

There are seven main groups of function types for the B/K 2032.

1. Key Control
2. Setup
3. Result I/O
4. Display I/O
5. Bus Control
6. Miscellaneous
7. Memory I/O

Step Response. The step response test requires the use of the Wavetek 172B Signal Generator and B/K 2032 System Analyzer. The generator is used to supply the step input to the closed-loop system while the B/K 2032 functions as an oscilloscope and monitors and stores the response of the SUT.

Since the B/K 2032 and Wavetek 172B are configured to a default condition after the performance evaluation option is selected, only deltas to this condition are required to be sent. The command requirements of the B/K 2032 and Wavetek 172B for each instrument are discussed in the remainder of this section.

Wavetek 172B Signal Generator. The magnitude and duration of the input step must be programmed into the signal generator. The Wavetek is commanded to change output

amplitude and pulse duration by sending a unique ASCII character sequence.

For example, to send a single step pulse of duration 10 seconds and amplitude 1 volt, the following sequence is sent.

'C6B1F.05A1.00JI<LF>'

where

C is Function identifier

6 is + Pulse function

B is Mode identifier

1 is triggered/single mode

F is Frequency identifier

.05 is .05 Hz which is a 10 second pulse (50% duty cycle of 20 second period)

A is Amplitude identifier

1.00 is 1.00 volts

J is trigger identifier to initiate pulse

I is Execute identifier

B/K 2032 System Analyzer. Approximate settling time given by the user determines the proper time base to be programmed into the instrument. The settling time is used as input to a subroutine which returns the Frequency Span setting required on the B/K to obtain the desired

sweep time.

The B/K 2032 is capable of generating sweep speeds such that frequencies down to about .0002 Hz are capable of being displayed on the CRT. The upper limit is about 25.6 Khz.

As an example, to send a command to the B/K to display a square wave of .1 Hz applied to channel A, the following command strings would be sent (assuming all default settings are programmed into B/K):

```
'ED FU,0<LF>'  
'EM TM,1<LF>'  
'EM TS,0<LF>'  
'EM TL,.25<LF>'  
'EM FS,xx<LF>'
```

where ED is the mnemonic for the function
Edit Display Specification

EM is the mnemonic for the function
Edit Measurement Specification

FU is the mnemonic for the field Function

0 is the value 0 for the Time Channel A function

TM is the mnemonic for the field Trigger Mode

1 is the code for Channel A trigger mode

TS is the mnemonic for Trigger Slope

0 is the code for + slope trigger

TL is the mnemonic for Trigger Level

.25 is the value of trigger level in fraction of maximum input level

FS is the mnemonic for Frequency Span

xx is the Freq Span which provides the proper time base

Frequency Response. This section generates the commands necessary to program the B/K 2032 System Analyzer perform a frequency response test. The B/K 2032 is used to generate a random white noise which is injected into the SUT. The frequency and phase characteristics of the output signal are then collected and displayed by the analyzer.

The user-supplied approximate bandwidth (in rads/sec) is converted to a value that is compatible with the B/K 2032. The frequency span of the B/K display is governed by the value supplied by the user. The B/K has a frequency range of 1.56 Hz to 25.6 Khz. The approximate bandwidth value must be converted to an integer format if greater than one, or to a decimal value if less than one. The frequency range is doubled and converted to the proper value.

As an example, consider the approximate bandwidth of a system to be 3.45 hz. This value is doubled to obtain 6.9 hz to allow sufficient frequency range to be observed.

Since the B/K 2032 is configured to a default condition after the performance evaluation option is

selected, only deltas to this condition need be sent. The command requirements of the B/K 2032 are discussed in the remainder of this section.

The command string for the above example is as follows :

'EM FS,6.9<LF>'

where

EM is the mnemonic for the function Edit_Measurement_Specification

FS is the mnemonic for the field Frequency_Span

6.9 is the value 6.9 hz for the frequency span which is rounded to the nearest achievable value by the B/K 2032

Level 6.3 Send Command String. A National Instruments GPIB11-2 IEEE-488 Interface Board was installed on the VAX 11/780 to give the system an instrument control capability (32). Software drivers to operate the board and provide interfaces with user programs were also installed on the VAX VMS system (33).

The command strings that have been formed to program the test instruments via the IEEE-488 bus must be sent down the bus to the appropriate instrument. The IEEE-488 user interface software provides an easy means to specify the

device number and the array of characters making up the command string.

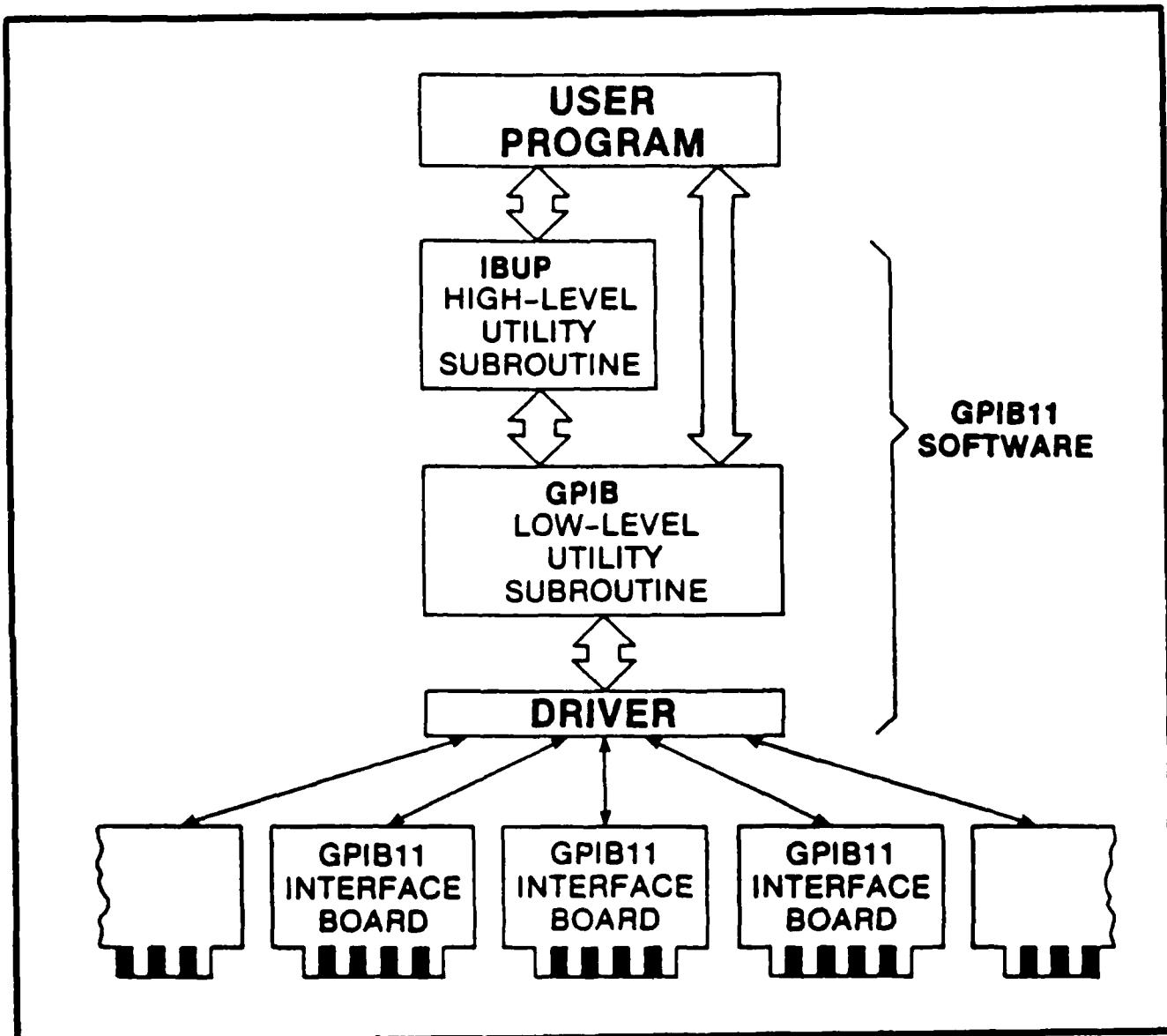


Figure 23. IEEE-488 Software Interfaces

IEEE-488 Bus Description. The IEEE-488 interface standard, also known as the General-Purpose Interface Bus (GPIB), offers a uniform method of sending parallel code from one device to another. The method of transmission is byte-serial, bit-parallel over a 24 line shielded cable which is extremely resistant to interference. There can be 15 devices on a single GPIB. The maximum length of cable connecting a group of instruments within a normal bus system is either two meters times the number of devices on the bus or 20 meters, whichever is less. A bus extender is used with DICES to extend the bus to 30 meters. The usual minimal system requires a system controller which decides which device on the GPIB is to communicate with any other device(s).

Devices on the bus may, in general, perform three main functions. They may be talkers, i.e., they may transmit data to other devices on the bus. Of course, there may be only one active talker at a time on the bus. Alternatively, a device may be a listener - it may receive data or instructions from another device on the bus. There may be more than one active listener on the bus at any given time. Lastly, a device may act as a controller, a coordinator of which device may talk and which devices may listen. A device may also simply stand by and do nothing.

The interface supports two modes of operation: command and data. As the name suggests, the command mode is

for commanding the instruments on the bus to listen, talk, or stand by. For example, on the Wavetek 172B Programmable Signal Source, the controller can command the 172B to listen for a message coming down the bus. In the data mode, data is transferred to the 172B which then interpreted as instrument-peculiar commands and data. The IEEE-488 standard does not dictate what the data means that is sent down the bus. It only specifies the format and protocol of the data that is transmitted on the bus. Each individual instrument interprets the data according to the particular manufacturers' conventions.

There is a sequence of detailed handshaking and signalling that goes on between each instrument and controller to transmit a word on the GPIB. This detail will not be discussed here but is available in the literature (31,38,39). The use of high-level interface programs avoid the need for the system designer to become entangled in the details of this process (33).

VAX 11/780 IEEE-488 Interface Description. A National Instruments GPIB11-2 interface card was installed on the ISL VAX 11/780. This card interfaces DEC UNIBUS-based computers to the GPIB. Figure 24 shows a typical implementation on a VAX 11/780 system.

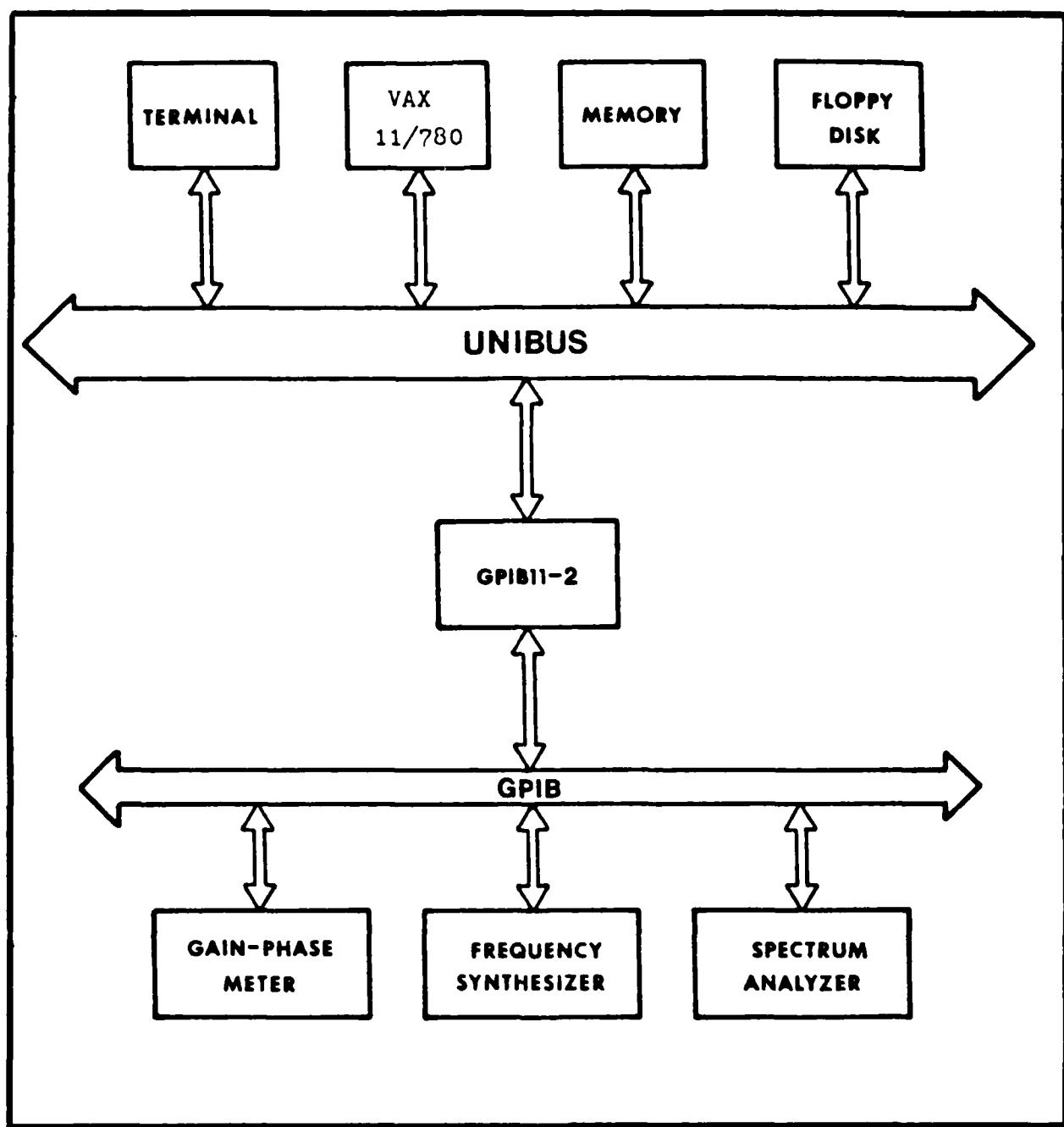


Figure 24. Typical VAX 11/780 GPIB System

This interface allows programming of instruments on the GPIB from within high-order-language programs, such as DICES.

During installation of the GPIB11-2 and VAX VMS software drivers, the GPIB device addresses are entered into the drivers. This allows reference to a device on the bus by simply specifying a device number such as 1, 2, or 3. The device addresses and numbers were configured as shown in the following table.

<u>Device</u>	<u>GPIB Address</u>	<u>Device Number</u>
GPIB11-2	25_8	0
B/K 2032	32_8	2
Wavetek 172B	00_8	3

Table 17. Device Numbers and Addresses

IEEE-488 Bus Extender. The IEEE-488 bus is limited to 20 meters total length with further limitations on distance between instruments. Because of the physical separation between the VAX 11/780 and the area where this work was carried out, a National Instruments GPIB-100 IEEE Standard 488-1975 Bus Extender was installed on the system. The GPIB-100 is capable of extending the IEEE-488 bus up to 300 meters. An extension of 30 meters was used for this investigation (See figure 25).

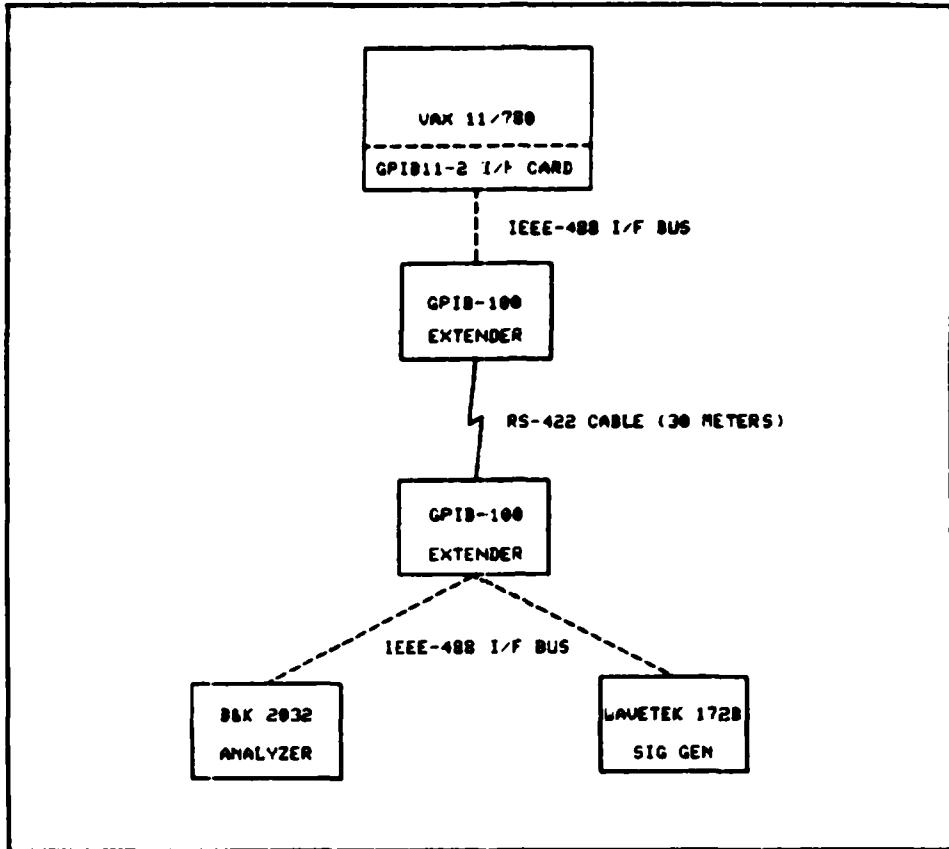


Figure 25. IEEE-488 Bus Extenders

High-Level Software Interfaces. There are several methods of calling the various GPIB functions, however, the only used within DICES is a FORTRAN function call IBUPU. This high-level utility function allows convenient programming by automatically handling the interface control protocol, such as device addressing associated with transferring device-dependent messages (device-unique commands and data) over the GPIB.

The IBUPU calling syntax contains the following parameters:

- o System Controller Board NUmber
- o Function Code
- o Zero or more arguments for a particular function

A brief summary of the IBUPU functions is given below.
For complete details refer to the GPIB11-2 VAX/VMS Software Reference Manual (33).

<u>Function</u>	<u>Code</u>	<u>Syntax</u>
-----------------	-------------	---------------

WRITE	0	J=IBUPU(0,0,D,ARRAY,COUNT)
-------	---	----------------------------

Addresses the selected device as Listener and transfers the data bytes from the computer to the device

READ	1	J=IBUPU(0,1,D,ARRAY,COUNT)
------	---	----------------------------

Addresses the selected device as Talker and transfers the data bytes from that device to the computer.

CLEAR	2	J=IBUPU(0,2,D)
-------	---	----------------

Selectively clears one device on the GPIB, simultaneously clears all devices on the GPIB, or initializes the GPIB itself by sending the Interface Clear (IFC) message.

TRIGGER *	3	J=IBUPU(0,3,D)
-----------	---	----------------

Selectively triggers one device or simultaneously triggers all devices.

REMOTE	4	J=IBUPU(0,4,D)
--------	---	----------------

Puts a selected device in remote mode, all devices in remote mode, or to put all devices in remote mode with local-lockout.

LOCAL 5 J=IBUPU(0,5,D)

Puts a selected device in local mode, all devices in local mode, or all devices in local mode and cancels the local-lockout condition.

POLL * 6 J=IBUPU(0,6,D)

Serially polls a selected device, parallels polls all devices, or tests if any device is requesting service.

CONFIGURE * 7 J=IBUPU(0,7,F,S,L)

Configures a selected device for parallel polling or to unconfigure all devices.

PASS CONTROL * 8 J=IBUPU(0,8,D)

Allows the selected device (with Controller capabilities) to control the GPIB.

DEFINE* 9 J=IBUPU(0,9,D,TAD,LAD,SAD,RMD,EOD,WMD)

Replaces the information in the bus table with the arguments specified in the call.

FINISH * 10 J=IBUPU(0,10)

Terminates usage of the GPIB.

where

D is the device number

ARRAY is a string of bytes to be written to the device

COUNT is an integer that gives the size of ARRAY

S is the sense of the response

L is an integer representing the data line for the response

TAD is an integer representing the GPIB Talk Address
(0100 - 0136 octal)

LAD is an integer representing the GPIB Listen Address
(040 - 076 octal)

SAD is an integer representing the Secondary Address
(0140 - 0176 octal)

RMD is an integer representing the Read Mode

EOD is an integer representing the End-Of-Data
character for certain read modes

WMD is an integer representing the Write Mode

J is an integer that represents the returned value
from the function call. A 1 is no error while negative
numbers are error codes.

* means function is not used in DICES

Error Codes. The IBUPU module returns error
codes to the integer variable being assigned the function
output (J in the previous syntax descriptions). These error
codes are very useful for troubleshooting and to determine
if the instruments on the bus have been properly connected
and turned on. A complete error code summary is given in
Appendix D of reference 33.

Level 6.4 Start Test Sequence. This process commands
the test instruments to begin the test sequence. This
involves turning on the generator output, and causing the
B/K analyzer to begin making measurements of the closed-
loop control system.

This process involves waiting for a user-generated signal to initiate the test sequence. A signal must be generated to start the sequence because there are some operations required to be performed by the user prior to starting the test. For all tests, the user must 'toggle' out of the transparent mode and communicate with the EVM to begin execution of the TMS32010 filter code. If the test sequence were started automatically, the user might not have sufficient time to ready the system.

The sequence of events that lead to the start of the test sequence are as follows:

1. Display message instructing user to start the TMS32010 executing the filter code
2. Read B/K for 'START' key
3. If not pressed yet - return to step 3
4. When pressed
 - A. Step Response - Start pulse output from Wavetek 172B
 - B. Freq/Phase Response - B/K is started by the actual closure of the 'START' switch
5. Return to Performance Evaluation Menu

For example, assuming the Wavetek 172B is set-up in the trigger mode to output a single pulse when triggered, the following code will generate the pulse.

```
PROGRAM TRIG  
INTEGER IBUPU  
CHARACTER*2 CMD
```

```
'PROGRAM TO TRIGGER 172B  
'DEFINE INTEGER FUNCTION
```

```
CHARACTER*2 CMD  
  
BOARD=0           'SYSTEM CONTROLLER #0  
  
FUNCTION=1        'WRITE FUNCTION CODE=1  
  
CMD='I'          'COMMAND STRING TO EXECUTE  
  
CMD=CMD//CHAR(12) 'ADD <LF> CHARACTER  
  
DEVNUM=3          'DEVICE NUMBER OF GEN  
  
LGTH=LEN(CMD)+1  'LENGTH OF CMD + 1 BECAUSE  
                  'OF ADDED <LF>  
  
J=IBUPU(BOARD,FUNCTION,DEVNUM,%REF(CMD),LGTH)  
  
STOP  
  
END
```

Where %REF(string variable) is used by VMS to pass
variable strings

Level 6.5 Collect Test Results. Collecting the test results when the test is completed is accomplished by using the cursor functions of the B/K 2032. These functions allow flexible read-out of the various test parameters, such as peak time, rise time, settling time, bandwidth, etc.

The B/K has several different cursor types which provide different graphical displays of the test response data. Upon completion of each test, the B/K is programmed to go to a particular cursor mode. This mode can be changed by the user via the front panel if another mode is desired.

The default mode selected by DICES is called the 'DELTA' mode. This allows setting a reference mark anywhere on the display (time axis or frequency axis) and measuring the delta time or frequency to a movable cursor. The value of the Y-axis parameter (Volts or dB magnitude) is also displayed as the cursor is moved. This is extremely useful to make the type of measurements that are usually of interest in a control system.

The upper right corner of the display is the cursor block on the display.

DELT	Y:	38.34 dB
	X:	2345Hz
	ΔX:	3423Hz

The initial reference point for the cursor is zero Hz or 0 seconds. To make a measurement, the user simply moves the cursor to the desired location and reads out the values of interest.

Hardware System Design

This section will assemble the hardware required to implement DICES. It will draw upon each of the previous discussions of the individual processes to be implemented.

The computer system to be used is the VAX 11/780 operating with the Virtual Memory System (VMS Version 4.2).

This system provides speed, adequate mass storage, and a good maintenance record to date.

User Interface

Since the user must interface with both the VMS operating system on the VAX and the monitor program on the EVM, it is necessary to provide a video terminal interface to the user for both systems. Also, since the VAX system will assemble the TMS32010 filter code and download it to the EVM, there must be a physical interface between the two systems.

The EVM provides two serial ports to enable connection to other systems. The EVM can be placed into a 'transparency mode' which allows the video terminal to communicate with the VAX and EVM by toggling back and forth between the 'transparency mode' and normal mode.

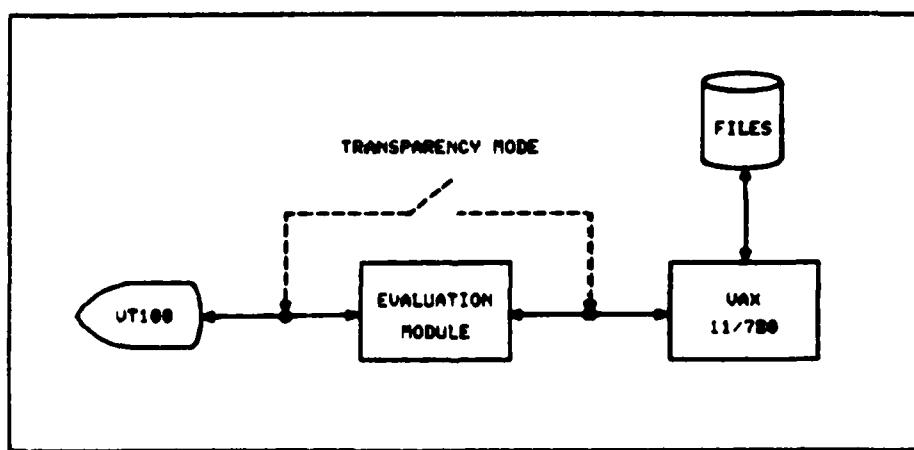


Figure 26. EVM/VAX Interface

This capability allows the user to interface with the EVM, while still maintaining the capability to interface with DICES and download the TMS32010 object code which will implement the controller design. This allows all DICES data data to be stored on the VAX and downloaded to the EVM when necessary.

Analog Computer Interface. The analog computer serves as the plant simulator, so it must be placed in the proper position to form the closed-loop system.

For a cascade unity feedback system, the closed-loop is as follows:

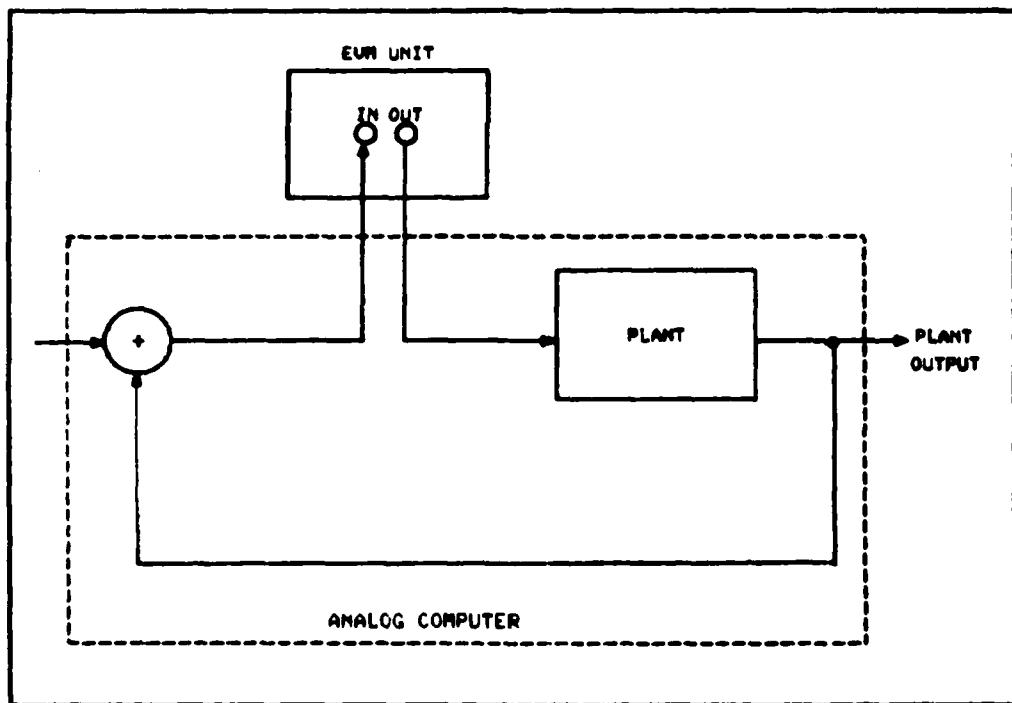


Figure 27. Unity Feedback Closed-Loop

For a feedback controller, the system appears as follows:

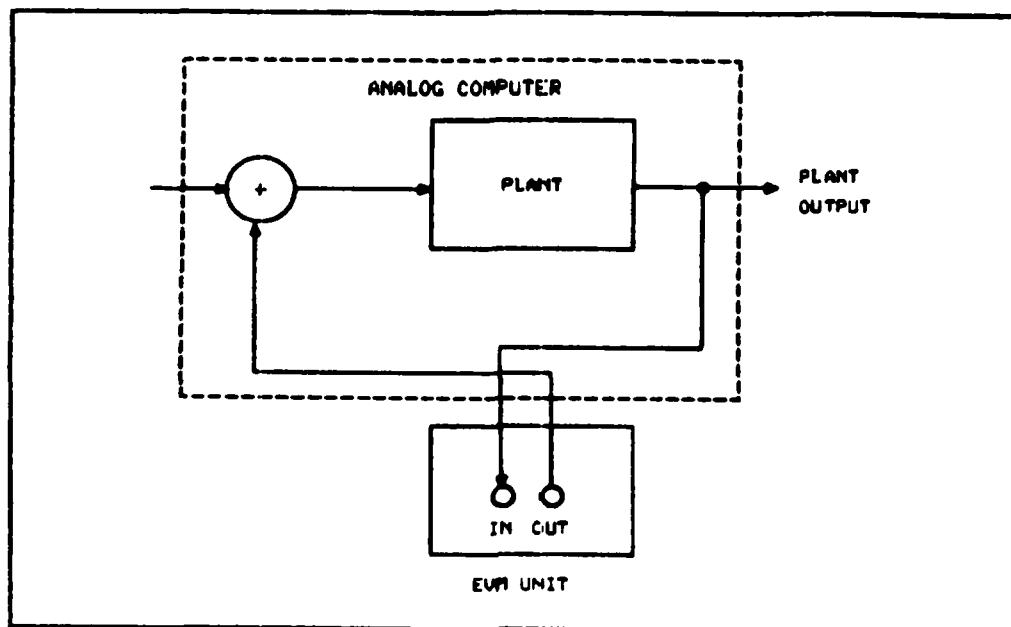


Figure 28. Feedback Controller Closed-Loop

In both cases, the plant is the combination of integrators, potentiometers, and amplifiers required to simulate the mathematical model of the plant.

Performance Analysis. To evaluate the performance of the system, there must be an input stimulus and something to measure or monitor the system output.

The input consists of the B/K 2032 System Analyzer and a programmable Wavetek Signal Generator. The B/K

contains a pseudo-random noise generator for frequency measurements and a narrow-pulse generator to simulate impulses into the SUT. The programmable signal generator is used to provide step inputs to obtain step-response information about the system.

The output of the system is monitored by the B/K System Analyzer.

The B/K Analyzer uses Fast Fourier Transform techniques to obtain frequency magnitude and phase information about the SUT. When operating in the time mode, the B/K 2032 acts as an oscilloscope to display the step response of the system-under-test. These data are stored in the B/K for further examination and study.

Combining all the system components produces the system block diagram as shown on the following page.

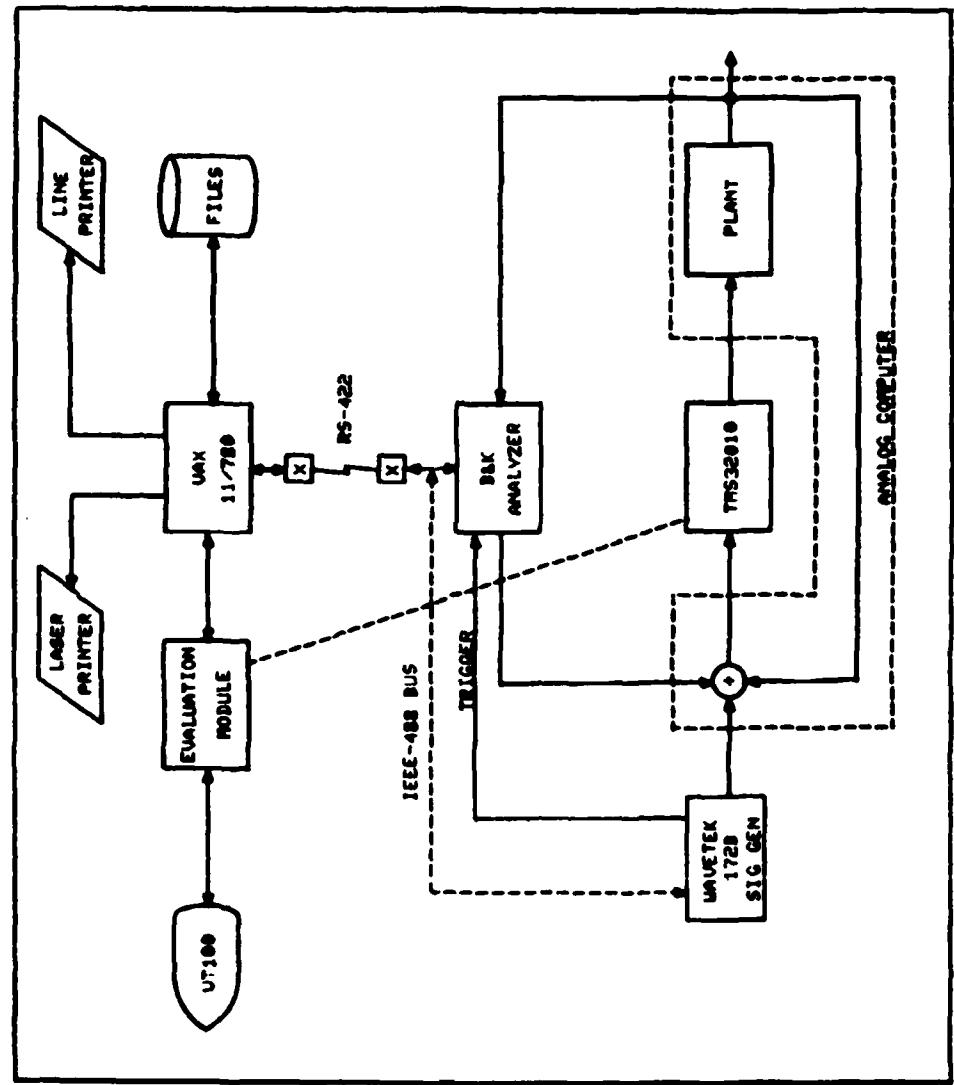


Figure 29. System Block Diagram

Summary

This chapter presented the detailed design of each major level within DICES. Each of the five major modules were designed based on the analysis performed for each module. Hardware and software designs were accomplished and detailed descriptions of each of the levels were given.

IV. - Implementation

Introduction

This chapter presents the implementation of the design of chapter 3 using various hardware and software modules. The approach taken is to address each of the various levels grouped according to the major modules identified in chapter 3. These modules are:

- 1. User Interface Module**
- 2. Filter Implementation Module**
- 3. Plant Modelling Module**
- 4. System Configuration Module**
- 5. Performance Evaluation Module**

The approach taken during the implementation phase is to implement each of the processes described in Chapter 3, and then use menus to obtain from the user the order of execution of each of the major modules. DICES is therefore actually a collection of modules, each of which performs a given function. However, as mentioned above, the user directs the order of execution of these modules and has control over which modules actually get executed.

System Implementation

Each of the five modules will now be implemented in the order shown above.

User Interface. This module contains the processes necessary to interface with the user to display and receive data. The entire module lends itself to implementation via software on the VAX VMS system. The processes to be implemented in this module are:

- | | |
|-----|------------------------------------|
| 1.0 | Determine system performance specs |
| 2.0 | Develop plant model |
| 3.1 | Display options |
| 3.3 | Test simulation |
| 3.4 | Connect system |
| 4.2 | Design controller |
| 4.3 | Analyze performance |
| 4.4 | Store coefficients |
| 5.1 | Retrieve coefficients |
| 5.5 | Re-analyze performance |
| 6.1 | Determine test requirements |
| 6.5 | Collect test results |

In addition, the main menu function will be implemented within this module. This routine presents to the user all the options available to the user.

Main Menu

The options available are presented to the user by

this module and the user is prompted to select the desired option. This code will be contained in the main program module called DICES.THS. DICES.THS will be the main calling program which displays the option menus and calls the appropriate subroutines. A listing of DICES.THS is provided in Appendix E.1.

The main menu and all the sub-menus will appear as follows:

DIGITAL INTERACTIVE CONTROLLER EVALUATION SYSTEM (DICES)

OPTIONS:

1. SYSTEM PERFORMANCE SPECS
2. MATHEMATICAL MODEL DEVELOPMENT
3. PLANT SIMULATION
4. CONTROLLER DESIGN
5. IMPLEMENT CONTROLLER
6. CLOSED-LOOP PERFORMANCE TESTING
7. REPORT GENERATION
8. END PROGRAM

Figure 30. Main Menu

CONTROLLER IMPLEMENTATION SUB-MENU (5)

OPTIONS:

1. READ DESIGN PARAMETERS AND PAIR POLES/ZEROS
2. SELECT QUANTIZATION ROUND-OFF OR TRUNCATION
3. FORM SECOND ORDER SECTIONS AND LOAD SOURCE CODE
4. GENERATE CONTROLLER OBJECT CODE (TMS32010)
5. LOAD OBJECT CODE INTO TMS32010
6. RETURN TO MAIN MENU

Figure 31. Filter Implementation Sub-Menu

PERFORMANCE TESTING SUB-MENU (6)

OPTIONS:

1. STEP RESPONSE
2. FREQUENCY RESPONSE
3. IMPULSE RESPONSE
4. RETURN TO MAIN MENU

Figure 32. Performance Testing Sub-Menu

Level 1.0 - Determine System Performance Specs

This process simply displays to the user a message to obtain system performance specifications. This message is contained within DICES.THS and called when option 1 on the main menu is selected.

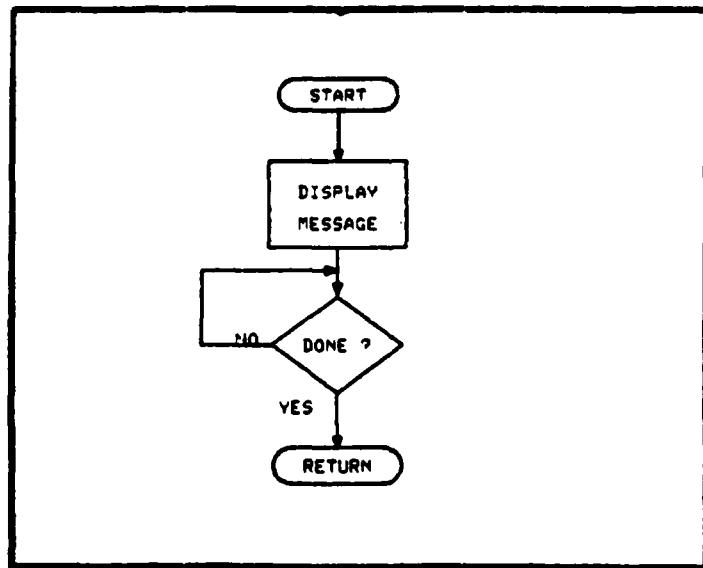


Figure 33. Levels 1.0 and 2.0 Flowchart

Level 2.0 Develop Plant Model

This process displays to the user a message to develop a mathematical plant model to be used in the design process. This message is contained within DICES.THS and called when option 2 from the main menu is selected.

Level 3.1 Display Assistance

This process displays to the user messages to assist in interfacing the analog computer, digital controller, system analyzer, and other DICES components. This process is called from the Plant Modelling Menu.

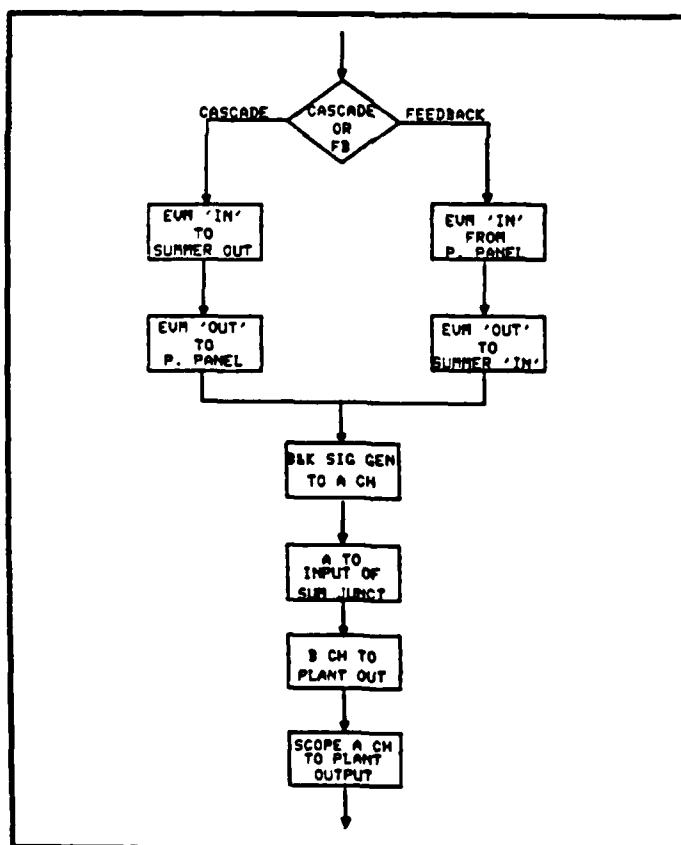


Figure 34. Display Assistance Flowchart

Level 3.3 Test Simulation

This process is simply the test of the plant simulation which is performed before the closed-loop system test is accomplished.

Level 3.4 Connect System

This level is the connection of the equipment which is performed by the user during level 3.1. No additional software is required for this process.

Level 4.2 Design Controller

This process allows the user to leave DICES and enter ICECAP for design of the controller.

When selected from the Design Controller Menu (level 4.0), this option simply exits the DICES main program into the ICECAP program. This is accomplished by use of the VAX VMS Runtime Library command 'SPAWN', which allows leaving one program, entering another, and returning to the first upon completion. The command within the FORTRAN program DICES.THS is as follows:

```
IST=LIB(dollar sign)SPAWN ('<at sign>[icecap1.modules]rice')
```

where IST is just a dummy integer variable.

Upon entering ICECAP, the user is free to design the controller for the plant model being used. Once the design is complete, the user copies the final controller design into the variable HTF and exits the program using the STOP

command.

Level 4.3 Analyze Performance

This process contains no code as it is performed entirely by the user using ICECAP and the appropriate design and analysis procedures.

Level 4.4 Store Coefficients

This process stores the compensator parameters when the design phase is complete. Upon issuance of the STOP command in ICECAP, control leaves the ICECAP program and saves the controller design in data file MEMORY.DAT. Instead of returning to the operating system, the user is returned to the DICES main menu.

Level 5.1 Retrieve Factors

This process retrieves the compensator parameters from the data file MEMORY.DAT. This module reads the sequential data file (MEMORY.DAT) until the appropriate data are encountered as described in the design chapter.

The data read by this module are the 1) order of the controller, 2) sample period, 3) pole and zero roots, and 4) loop sensitivity gain.

This function will be broken into two modules called RDORDER.THS and RDHTF.THS. RDORDER.THS reads the order of the controller and RDHTF.THS reads the remaining data.

Controller order must be read first from the data file because it is located after the other data in the file. This allows apriori knowledge of the number of poles and zeros the second time through the sequential data file.

The flowcharts for RDORDER.THS and RDHTF.THS are shown below.

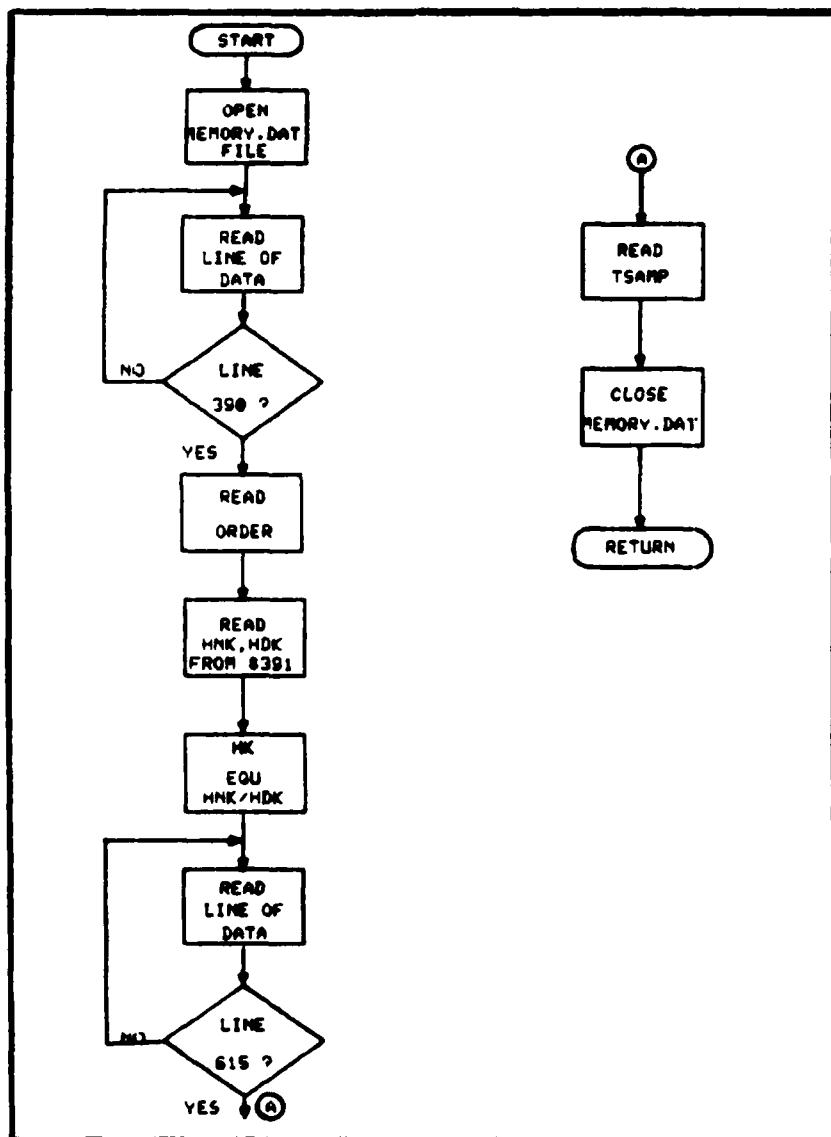


Figure 35. RDORDER.THS Flowchart

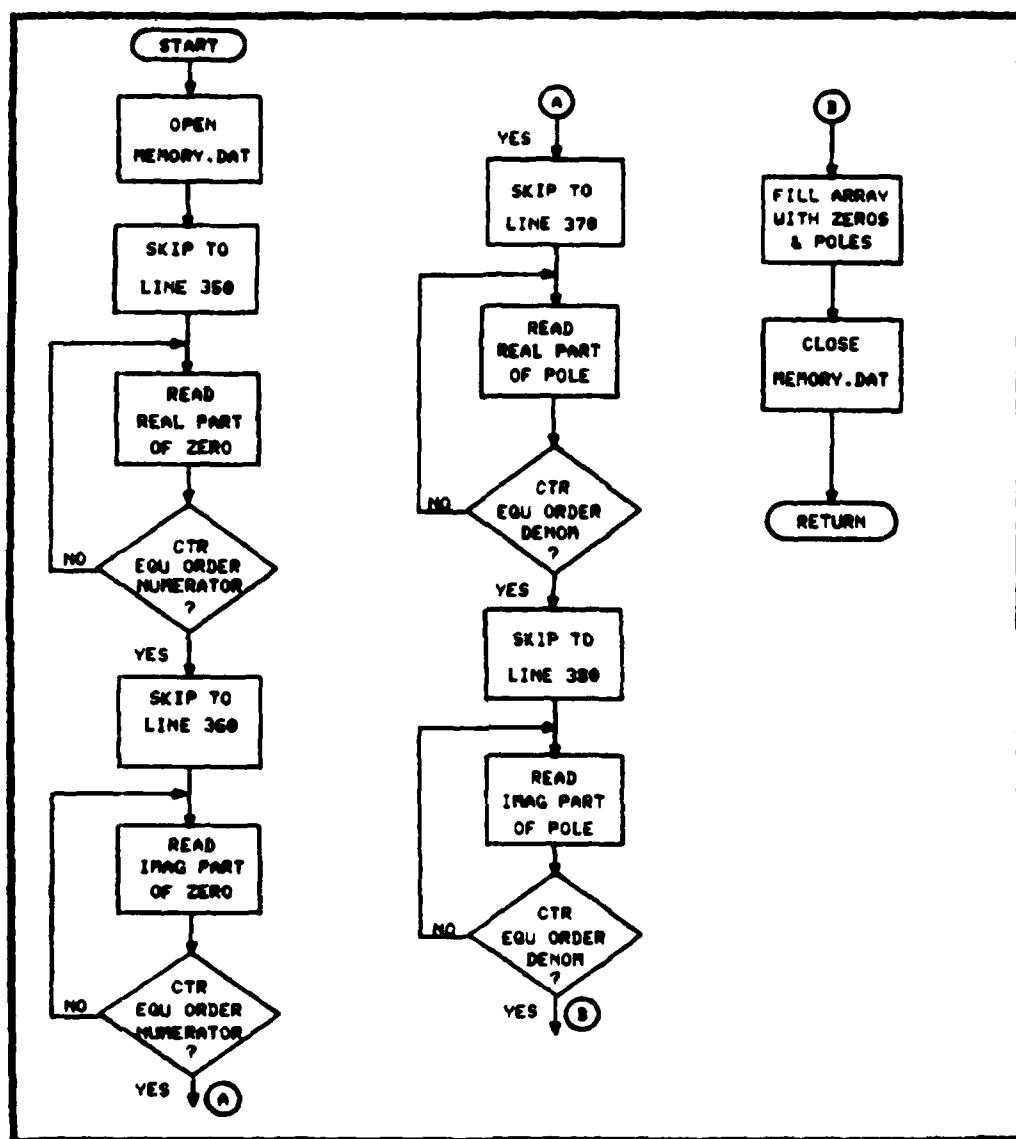


Figure 36. RDHTF.THS Flowchart

Level 5.5 Re-analyze Performance

This module will place the quantized poles and zeros of the controller design back into the MEMORY.DAT data file for analysis using ICECAP.

Once ICECAP is entered, a RECOVER (29) command is used to retrieve the data from MEMORY.DAT. An analysis can then be performed to determine the theoretical effects of quantization on the performance of the closed-loop system.

Since ICECAP does not refactor the numerator and denominator polynomials when a RECOVER is performed, DICES must multiply the separate first and second-order sections together to form the full-order polynomials. The poles and zeros of the quantization process must be obtained by factoring the second-order sections. These factors must be reinserted into the MEMORY.DAT file. This function is not currently available in DICES.

Level 6.1 Test Requirements

The user must make known to DICES the tests to be performed on the system by selecting the Performance Test Option from the main menu, and then the specific test from the Test Menu. The current version of DICES implements only the Step Response and Frequency (Magnitude and Phase) Response Test. The Impulse Response Test is not implemented.

Main Menu Option 6 is selected to begin the test phase and the Test Menu is displayed (See figure 32). Once the

selection is made, the test-type is passed to the next process.

Level 6.5 Collect Test Results

Following completion of a particular test, the results must be made available to the user for review. The B/K 2032 is used as the primary means of displaying the test results. Following either a Step or Frequency Test, the B/K 2032 processes the system response and displays the results in a meaningful manner.

Step Response Test. The Step Response Test display consists of a full-screen format display of the time response of the system-under-test. The x-axis represents time while the vertical axis represents volts (which represent some physical unit after conversion). The time axis scale was set according to the user-provided settling time of the system during the test set-up. Vertical scaling was initially set according to the specified step amplitude but following test completion, is set to maximum resolution using the AUTOSCALE function of the B/K 2032 (40). When the TEST COMPLETE message appears at the bottom of the B/K, the user is free to use the FIELD ENTRY knob on the B/K panel to move the cursor to determine system response parameters of interest.

Frequency Response Test. Frequency Response testing

produces a split-screen format which contains both the magnitude and phase response. The upper screen displays the magnitude response with the horizontal axis representing frequency (in Hertz), while the vertical axis is calibrated in decibels. The lower screen displays the phase response of the system. The horizontal axis is identical to the upper magnitude display while the vertical axis is in degrees. As with the Step Response Test, the user is free to move the cursor about the display. The default mode for the cursor is ALIGN CURSORS, which moves the cursor on both displays simultaneously. Consult reference 40 for additional cursor modes.

Filter Implementation

This module contains the modules necessary to implement the digital filter design on a TMS32010 microprocessor. The processes contained within this module are listed below.

- 5.2 Select filter structure
- 5.3 Pole/zero pairing
- 5.4 Quantize coefficients
- 5.6 Scaling and ordering
- 5.7 Sample period selection
- 5.8 Generate TMS32010 code

Level 5.2 Select Filter Structure

This module permits the user to select the structure of the digital filter to be implemented on the TMS32010. The current version of DICES implements the 3D structure, however, future versions could allow any of the standard structures to be used.

The selection of the filter structure is accomplished by choosing the 'Select Filter Structure' option from the Filter Implementation menu. The user is then presented with another menu which lists the current structures available.

A generic eighth-order 3D filter is contained in data file 3DFILT.THS. For future versions of DICES, it is suggested that separate data files each containing a filter structure source program be maintained. For example, a program which implements a 1D structure would be called 1DFILT.THS.

3DFILT.THS Implementation. This section discusses the details of implementing the generic 3D filter program. The basic design of the filter program was taken from a Texas Instrument's Digital Signal Processing Application Report entitled, 'Implementation of FIR/IIR Filters with the TMS32010' (4:C7-C9). A complete program listing is contained in Appendix E.2 which should be used to complement the following discussion. The line numbers refer to the leftmost numbers that have been assigned by the assembler.

Lines 75 - 141. This section equates variable names to integer values which will used as memory location labels.

Line 144. Data location zero is used as the starting location for assembly of the source code.

Lines 168 - 195. This section uses DATA statements to define the coefficients of a unity gain filter. These coefficients are modified by DICES depending on the poles and zeros of the filter to be implemented. This section also initializes the sample period, Analog Board mode, and the constant 400. MASK1 and MASK2, which are used to convert various number codes, are also initialized.

Line 200 is the beginning of the executable code.

Lines 202 - 242. This section moves the data constants from program memory to data memory for access by the TMS32010. In the TMS32010 architecture, constants must reside in data memory to read into the ALU. A special instruction called TBLR (Table Read) exists to move data from program memory to data memory.

All the initial states of the delay nodes of the filter are initialized to zero. An initialization word is sent out to the Analog Interface Board as well as the sample period constant.

Line 245. The beginning of the infinite loop which executes the difference equation.

Lines 248 - 254. This section reads the input signal

from the Analog Interface Board at a rate determined by the sample period constant that was loaded. Because of the hardware limitation on the sample periods that can be obtained, the input is kept only every 400th sample. This effectively scales the range of sample-periods by a factor of 400.

Lines 258 - 260. A format conversion from the AIB's Complementary-Offset-Binary (COB) code to Twos-Complement.

Lines 262 - 280. Performs the multiplications, additions, and data shifts of the input and output samples to implement the first section of the filter. This section relies heavily on a unique instruction pair called LTA and LTD of the TMS32010 which has been optimized for digital filter applications. These two instructions, in conjunction with a MPY (multiply) instruction:

1. Perform the summation of a previous partial product
2. Load an input or output data value into the T-register as a multiplicand
3. Shift the data point being loaded by one location which represents a 'delay' of one sample-period
4. Multiply the T-register by the coefficient for the particular input/output data value

Lines 281 - 297. Implements the second section with code similar to section one.

Lines 298 - 314. Implements the third section with

Lines 298 - 314. Implements the third section with code similar to section one.

Lines 315 - 331. Implements the fourth section with code similar to section one.

Lines 334 - 337. Converts from Twos-Complement format to Offset-Binary (OB) format for the D/A converter.

Line 339. Outputs the computed output value, $Y(k)$, to the AIB for control of the plant.

Line 341. Branches to beginning of the infinite filter loop at line 248.

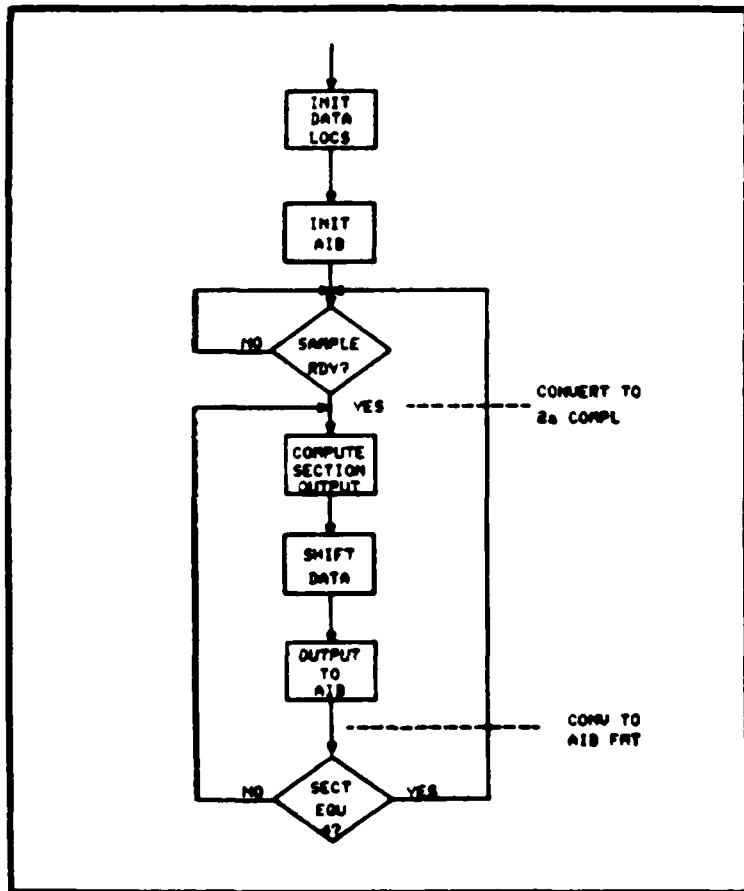


Figure 37. Flowchart of 3D Filter Implementation.

Level 5.3 Pole/Zero Pairing

The pole/zero pairing module receives an array of filter poles and zeros and selects the pairs of poles and zeros that will make up each second-order section of the filter. In cases where the order of the denominator is odd, there will always be a first-order section.

The algorithm to be implemented consists of subroutines that are called from the main routine PAIR.THS. This main routine is basically an iterative routine that first pairs the poles and then assigns zeros to each section until the supply is exhausted. Since the order of the numerator is less than or equal to that of the denominator, there may not be sufficient zeros to assign to each section. A numerator of 'one' is simply used in these cases.

Implementation of the pairing algorithm is flowcharted in figures 38 and 39. This routine calls some or all of the subroutines, depending on the exact constellation of poles and zeros for a particular filter. Appendix D details several different test paths through the flowchart, along with the resultant pairings that were generated.

Subroutines. Each of the main subroutines is now discussed along with flowcharts where warranted.

ANYCOMPZEROS. This routine determines if there are any complex zeros available for assignment to a filter

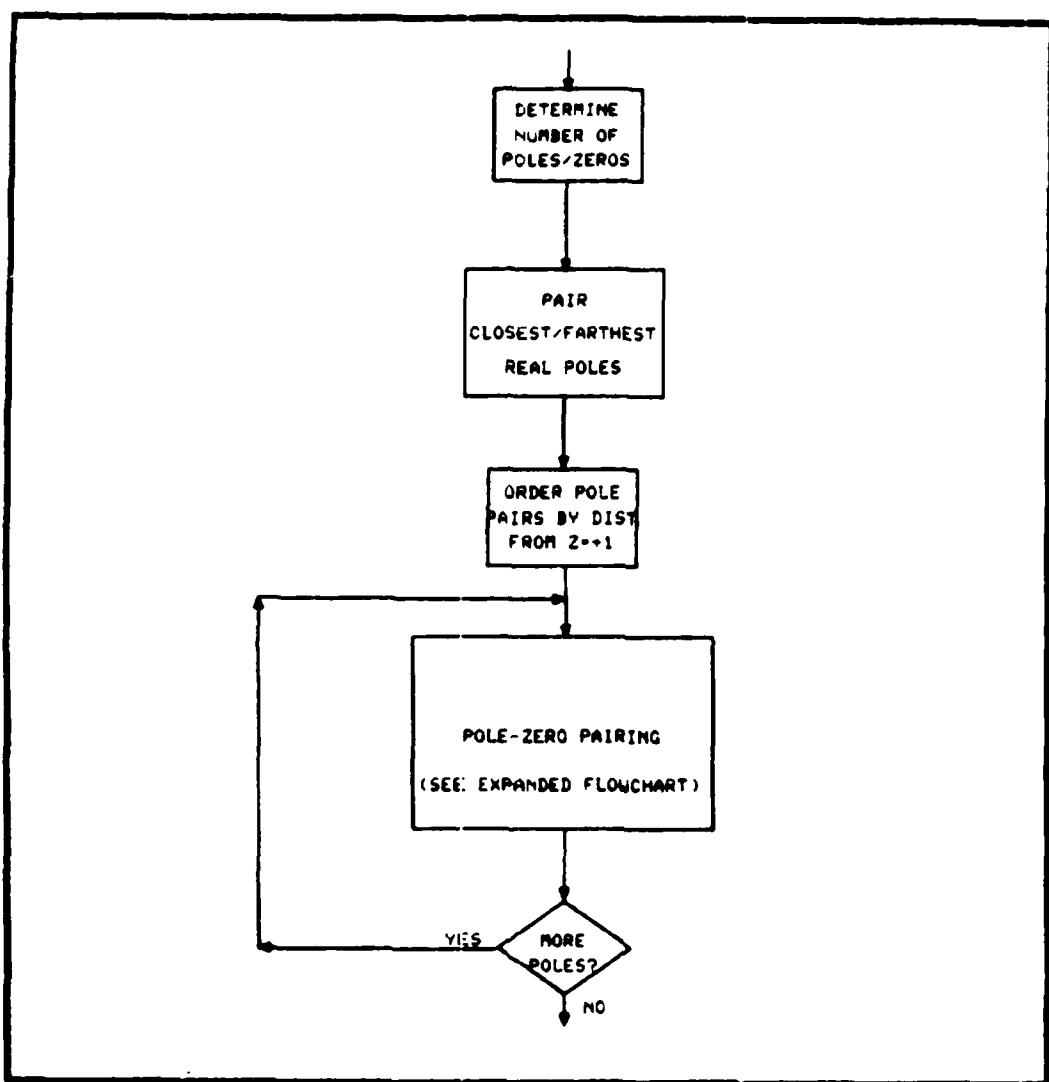


Figure 38. PAIR.THS Flowchart

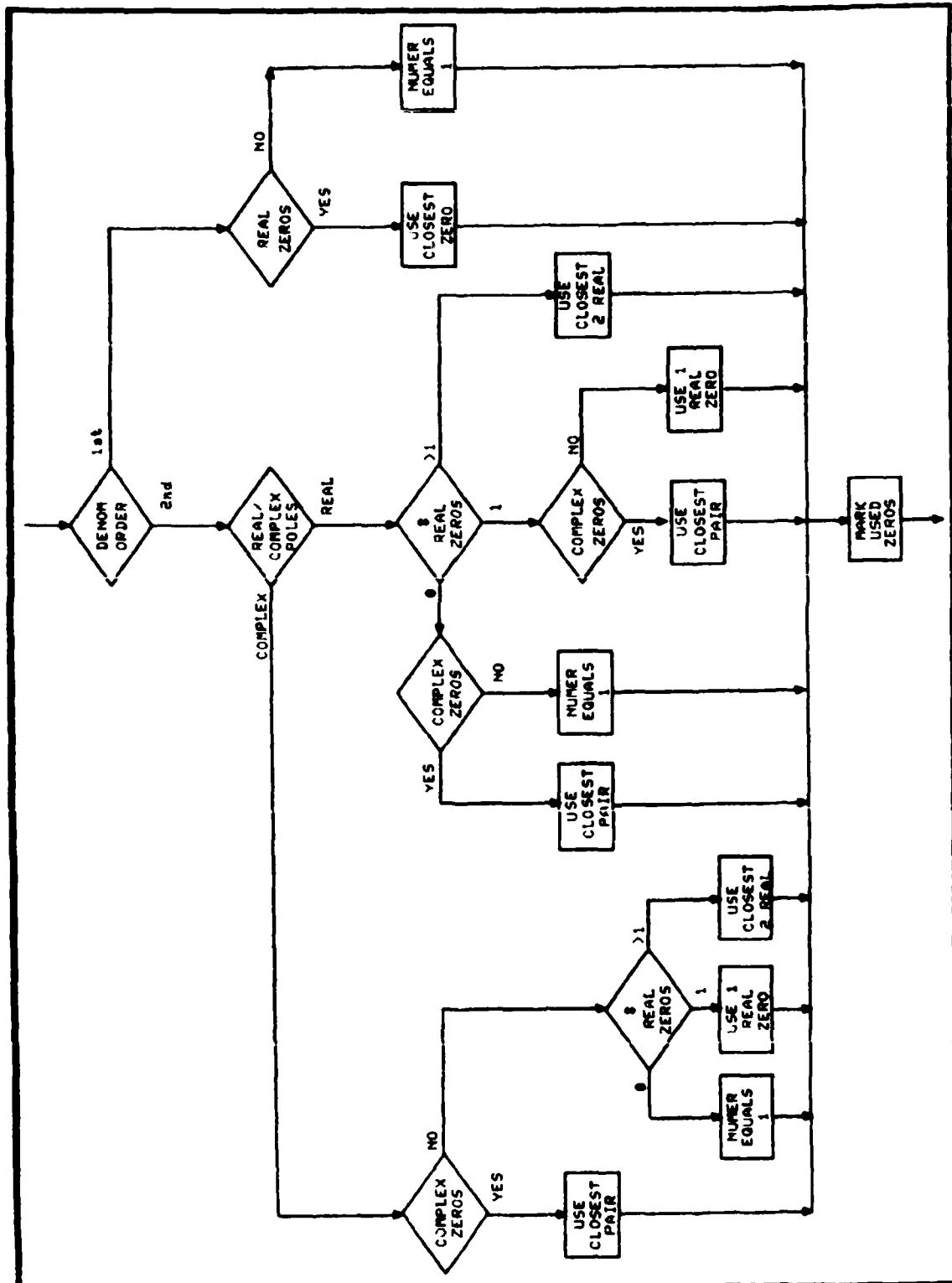


Figure 39. PAIR.THS Flowchart

section. 'Available' means that there is at least one pair of complex zeros that can be assigned to a filter section. This pair is then marked as 'used' so as to remove it from further consideration.

The input to this routine consists of a complex array of all the zeros of the compensator in no particular order except that complex zeros are adjacent to each other. This array is called CZERO. There is also a vector that initially contains the numbers one through m (where m is the order of the numerator), which represent the zero positions in the array. As each zero is used, a zero is placed in the corresponding vector location to represent that zero being unavailable for the rest of the pairing process. This vector is called IUSEDZEROS.

The routine simply iterates through the array 'm' times looking for a non-zero imaginary component of the array. When one is found, another is assumed next to it and the vector is assigned a value of '1' in those locations. This is continued until all zeros have been checked.

The routine returns the vector IAVAILZEROS which indicate the indices of complex zeros in array CZERO. The number of complex zeros available is also returned in variable INUMAVAIL.

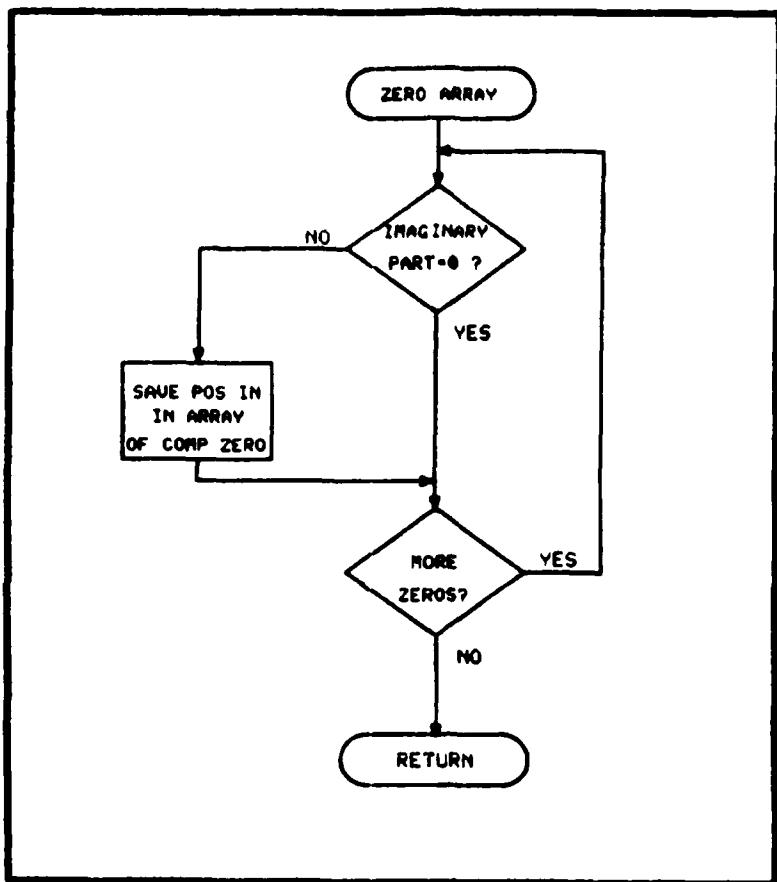


Figure 40. Subroutine ANYCOMPZEROS

ANYREALZEROS. This routine is similar to ANYCOMPZEROS except it checks for the imaginary part of each zero equal to zero. If it is and it has not been used yet, it is recorded in vector IAVAILREALZ. The number of real zeros available is passed out of the routine in variable INUMAVAIL.

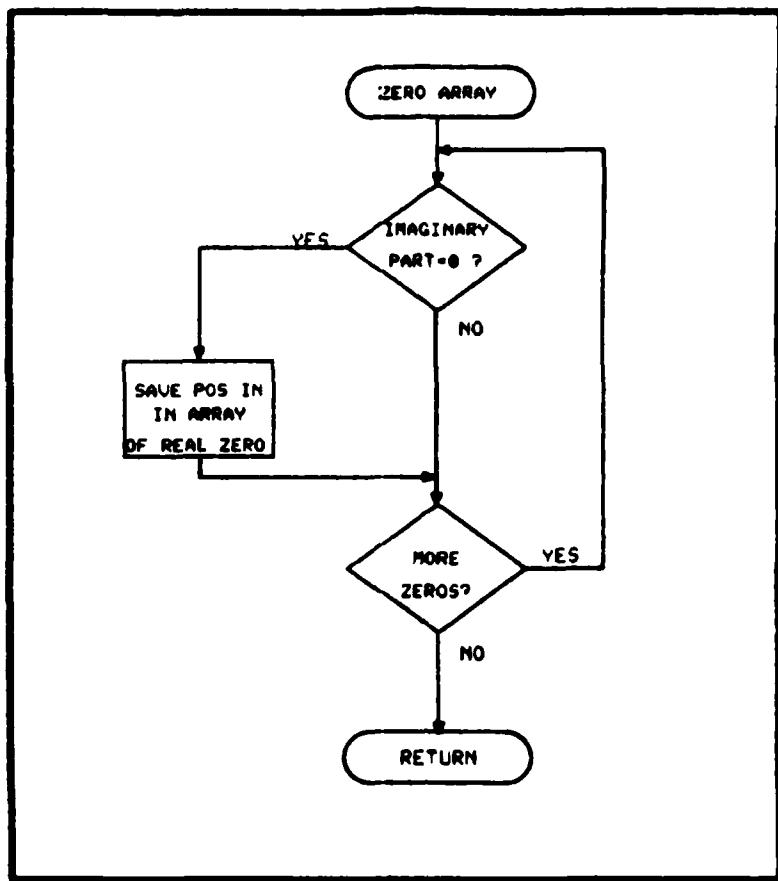


Figure 41. Subroutine ANYREALZEROS

MARK1Z. This routine records the use of a real zero in the pairing process. The inputs to this routine are IUSEDZEROS and ICLOSER1. The output is IUSEDZEROS that has been appropriately marked.

IUSEDZEROS has been described in subroutine ANYCOMPZEROS. ICLOSER1 contains the index number of the zero to mark in IUSEDZEROS.

IUSEDZEROS is returned with the latest used zero marked to remove it from further consideration in the

pairing process.

MARK2Z. This routine is identical to MARK1Z except two zeros are marked instead of one. The zeros to be marked are passed to the subroutine by the variables ICLOSERPAIR or ICLOSECPAIR, depending on whether the pair is real or complex.

CLOSEST1ZERO. This routine locates the closest single zero to a reference pole that is passed to the subroutine. The inputs are CZERO, CPOLEPAIR, IAVAILREALZ, and INUMAVAIL.

CZERO is the array of zeros.

CPOLEPAIR is the reference pole.

IAVAILREALZ is a vector of index numbers that locate the real available zeros.

INUMAVAIL is the number of zeros available for consideration.

The output is the variable ICLOSER1, which contains the index number in CZERO of the closest real zero to the reference pole.

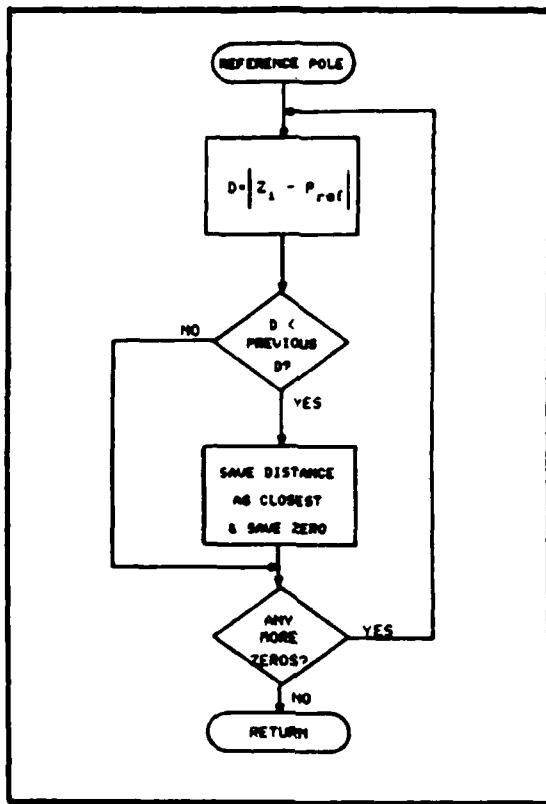


Figure 42. Subroutine CLOSEST1ZERO

CLOSESTREALPAIRC. This routine is similar to CLOSEST1ZERO except that the reference pole is assumed to be a complex pair. The input variables are the same as CLOSEST1ZERO. The output variable is ICLOSERPAIR, which contains the closest two real zeros to the reference complex pole.

CLOSESTCOMPLEXPAIR. This routine locates the closest pair of complex zeros to the reference complex pole pair. The inputs are the same as CLOSESTREALPAIRC except

for the variable IAVAILCOMPZ which contains the index numbers of available complex zeros. The output is ICLOSECPAIR, which identifies the index number in CZERO of the closest complex pair of zeros to the reference pole.

CLOSESTREALPAIRR. This routine locates the closest real pair of zeros to a reference pair of real poles. The inputs and outputs are the same as CLOSESTREALPAIRC.

Level 5.4 Quantize Coefficients

This level multiples the second-order section roots to obtain first and second-order polynomials. The coefficients of these polynomials are then quantized into a 16-bit integer representation and stored in a data file for later use. The integer equivalent of the sample-period is also stored in the data file.

A 21 element array INEWCO contains the integer coefficients that are computed from the routine. It also contains the integer representing the sample-period, ITSAMP.

The routine assigns the integer 32767 to the b_0' (see page 91) coefficients as a default value. This provides a near-unity gain for unused sections of the filter. The actual gain of the section is

$$A = \frac{32767}{32768} \quad (44)$$

$$A = .9999694 \quad (45)$$

This attenuation is compensated for in the sections that are used by increasing the loop sensitivity gain factor by $(.9999694)^{-k}$ where k is the number of unused sections. This gain is then distributed over the used sections.

Example:

If two sections are used to implement a third or fourth-order filter with an $HK=.34345$, there will be two unused sections. These two sections attenuate the gain by $(.9999694)^2$ or $.9999388$. This requires a correction factor of 1.000061203 to be distributed over two sections. Each section's coefficients will be increased by a factor of $(1.000061203)^{.5}$ or 1.000030601 multiplied by the portion of HK assigned to that section. In this example, the HK for each used section would be $HK'=(.34345)^{.5}$ or $.586046073$. HK' is then multiplied by 1.000030601 to yield $HK''=.586064007$.

Each used section's roots are multiplied out to obtain high-precision coefficients and then multiplied by its correction factor. These coefficients are then quantized into 16-bit integer representations as described in chapter 3.

As each set of coefficients is formed for each section, they are stored in the array INEWCO. When the entire process is complete, the coefficients and ITSAMP are written to data file NEWCO.TXT for later access by subroutine CHNGCO.THS, which reads the coefficients and inserts them into the source code for the generic filter (3DFILT.THS).

This subroutine also stores the high-precision and quantized coefficients in a data file PAIR.PRT which also contains the filter sections from subroutine PAIR.THS. This data file may be printed to obtain a hardcopy listing of the coefficients and filter sections.

Level 5.6 Scaling and Ordering

This level is responsible for scaling the coefficients of the compensator so as to minimize the probability of overflow. The current version of DICES addresses this by only checking for coefficients that exceed a value of $\pm .9999694$ so that when quantized, they do not cause integer values greater than ± 32767 . When this occurs, the user must factor a portion of the gain out of the filter and

implement it in the analog computer system. Consult reference 13 for an excellent treatment of scaling and ordering.

Level 5.7 Sample Period Selection

This module allows the user to select the sample-period desired or have DICES extract the sample-period used during the design phase from the ICECAP MEMORY.DAT data file. The default value will be the value contained in the data file, as this is the normal design situation.

Level 5.8 Generate TMS32010 Code

This module generates TMS32010 object code from the source file 3DFILT.THS. This object code is the final digital filter code that is ready to be loaded into the microprocessor.

The TMS32010 Cross Assembler is resident on the VAX VMS system. To call this assembler from DICES, the user library command SPAWN is used. After completion of the assembly process, the VMS system returns the user to DICES.

This function is performed in the main module DICES.THS as a simple call to the assembler using SPAWN. The assembler then requests a filename which contains the source code to be assembled. The file 3DFILT.THS contains the source code for the digital controller designed via DICES.

Plant Modelling

The plant modelling module involves wiring the analog computer system to produce the model of the plant and directs the connection of the plant into the closed-loop control system.

This module consists of the following levels:

3.2 Patch-Up Equations

3.4 Connect System

Level 3.2 Patch-up Equations

When option 3 of the main menu is selected, instructions are given to wire the plant model using the analog computer system. This module simply gives some suggestions and references to assist the user in 'patching-up' the equations.

Level 3.4 Connect System

This module gives information on the connection of the equipment that makes up DICES. The assistance is given for each of the major items in the system. The DICES' Users Manual should be consulted during the system configuration.

Option 3 provides a step-by-step connection description for configuring DICES. It is assumed that the EVM is connected properly between the VT100 terminal and

the VAX VMS system. The users manual should be consulted to accomplish this minimum system configuration to perform the remaining steps. An outline of the remaining steps follows:

1. IEEE Interfaces
 - A. IEEE Bus Extender
 - B. Test Equipment
2. B/K 2032 System Analyzer
3. Wavetek 172B Programmable Signal Generator
4. TMS32010 EVM Analog Input/Output

Each of these steps will be displayed as a full page help display on the VT100. The user will also have the option of exiting the HELP process at any point and return to the main menu.

System Configuration

This module configures the various system components. It initializes equipment items that require start-up commands and also loads TMS32010 object code into the TMS32010.

The System Configuration Module consists of the following levels:

4.1 Access CAD Package

5.9 Load TMS32010 Object Code

Level 4.1 Access CAD Package

This module permits the user to access the ICECAP design CAD package on the VAX VMS system. This is accomplished by use of the SPAWN command from the VAX VMS user's library. This command allows exiting DICES to the ICECAP program. When ICECAP is exited, instead of returning to the VMS operating system, the user is returned to the DICES main menu.

Level 5.9 Load TMS32010 Object Code

This module permits the user to load TMS32010 object code from the assembly process into the TMS32010 microprocessor for execution.

When this option is selected, the user is returned to the VMS operating system. From VMS, the user must enter the 'non-transparent mode' of the EVM by sending the control character reserved for that purpose. This puts the user in contact with the EVM monitor program. Using the 'Load Program Memory' command, the EVM is readied for accepting object code from the VAX VMS system. The user then toggles the system back to communicate with the VAX and the object code is then dumped to the EVM. The details of this step are contained in the DICES Users' Manual (Appendix A) and

should be consulted to perform this step of the implementation process.

Upon completion of the loading process, the user is in communication with the EVM monitor. The user toggles out of this mode and when in contact with VMS, issues the command to execute DICES again.

Performance Evaluation

This module contains the routines required to program, start, and collect the resulting test data from the instruments. The commands to perform the test sequence are generated based upon test parameters that have been specified by the user.

This module consists of the following levels:

- 6.2 Generate Command String
- 6.3 Send Command String
- 6.4 Start Test Sequence

A general flowchart of the test sequence is also included to clarify the entire test execution phase.

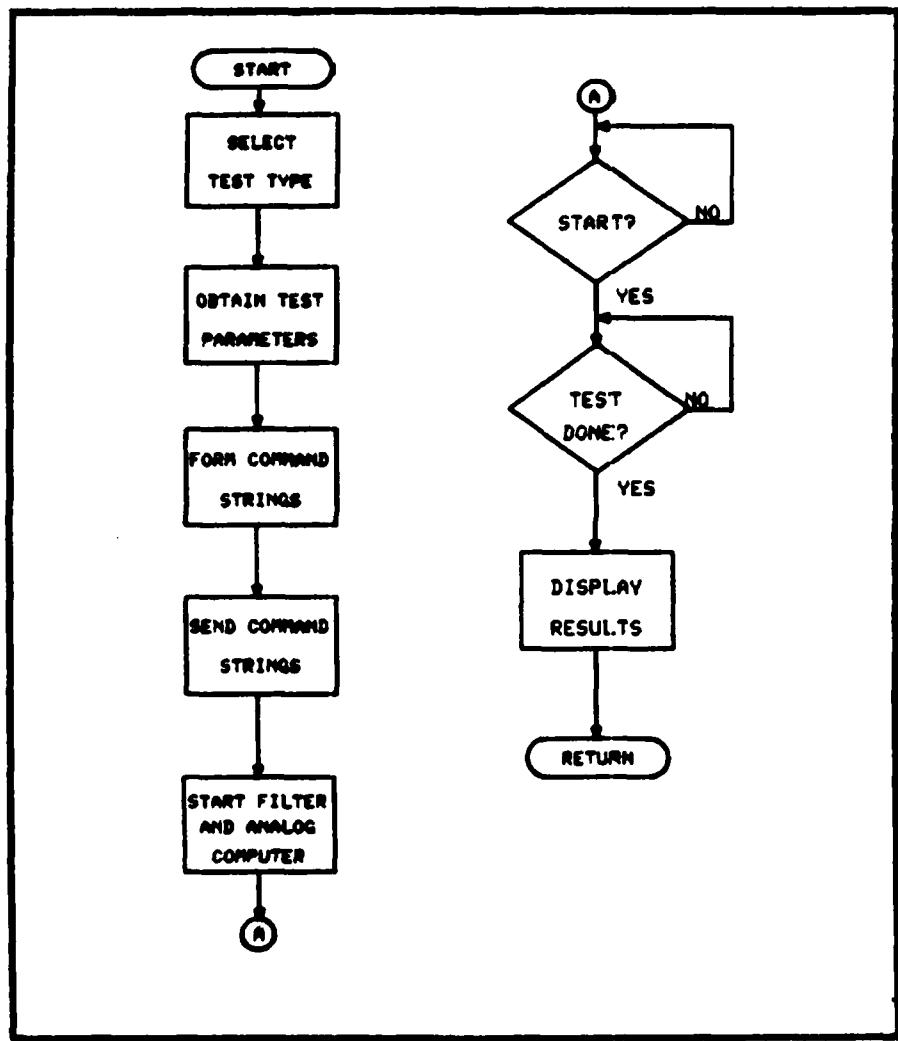


Figure 43. General Test Sequence Flowchart

Level 6.2 Generate Command String

After the tests to be performed have been determined, it is necessary to generate the command strings necessary to program the test instruments that will perform the tests. As discussed in Chapter III, the command strings consist of ASCII character strings that are sent to each instrument. Each instrument has its own manufacturer-unique response to a particular character string.

In DICES, there are two test types that can be performed - Step Response and Frequency Response. Each has a certain set of functions required from the Wavetek 172B and B/K 2032. Each test type will be addressed and the command string development for each will be discussed. Chapter III for complete details of the command structure for both instruments.

Step Response Test. The Step Response Test requires the use of both the Wavetek 172B and B/K 2032 System Analyzer. The 172B generates the step input signal to the closed-loop system and the B/K 2032 functions as a storage oscilloscope to record the system response versus time.

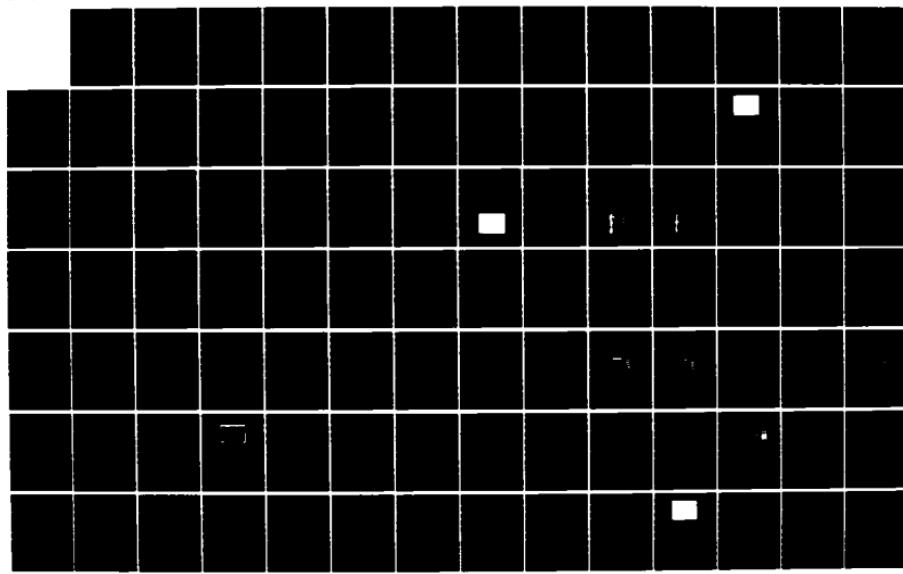
Wavetek 172B. The Wavetek 172B must be programmed to output a pulse that has the amplitude and duration requested by the user. The amplitude, A, is obtained directly from the user as a response to a prompt which requests the step amplitude. The user next inputs the approximate settling time, T_s , of the system and this value is used to enter a wait-loop before the amplitude is set to

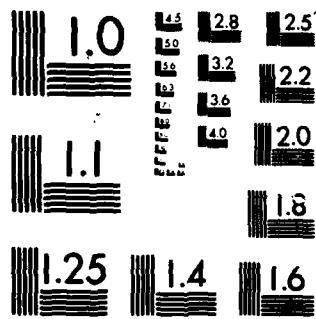
AD-A163 966 DEVELOPMENT OF A DIGITAL INTERACTIVE CONTROLLER
EVALUATION SYSTEM (DICES)(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
UNCLASSIFIED S B ECKERT DEC 85 AFIT/GE/ENG/85D-13

3/5

F/G 9/5

NL





MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS 1963-A

zero volts. The command string must cause the 172B to output a voltage of A volts for T_s seconds.

The Offset DC mode is used to obtain a DC output which is switched on when the proper command is received. Since the command string must consist of ASCII characters, a subroutine is used to convert the numeric data to character data.

The final command string appears as follows:

'D1.25B0C4P0I'

where the 1.25 represents the amplitude of 1.25 volts as a character string '1.25'.

B/K 2032. The B/K must be programmed to the Time Function mode for the step response test. Parameters such as Frequency Span (which sets the sweep time) and vertical sensitivity must be set. There are several less obvious parameters that must also be set that are only listed at the end of this section. These commands are straightforward and documented in the code.

The B/K is first set to operate in the Time Function Mode, which basically puts the Analyzer into an oscilloscope mode. This is accomplished by the following command:

'ED FU,1' ==> 'Edit_Display_FUnction, 1'

To set the Frequency Span correctly, a look-up table is used to convert from T_s to Frequency. For example, a T_s of 17 seconds is converted to a character string of '50', which is put into the command string.

'EM FS,50' ==> 'Edit_Measurement_Frequency_Span,50'

The vertical sensitivity is set by sending the amplitude, A, plus an additional .5 volts to allow for overshoot. This value is not critical, as the AUTOSCALE function is used at the conclusion of the test to optimize the display. The command string is as follows:

'ED YF,1.75' ==> 'Edit_Display_Y_Fullscale,1.75'

where the 1.75 represents 1.75 volts full-scale.

Additional Parameters. There are many other parameters that must be set to perform the step test, but only a list is given here. The source code should be consulted for the details of each command.

- Full Display Mode
- Cursor Mode Set
- Generator OFF

- External Triggering
- Positive Slop Trigger
- Single RECORD Mode

Frequency Response Test. The Frequency Response Test requires the use of only the B/K 2032 System Analyzer. This instrument generates the required stimulus and also performs the data analysis to obtain the frequency response.

B/K 2032. The B/K must be programmed to a pre-defined function (Frequency Response) available on the B/K. This function determines magnitude and phase response of a system and displays the results. The main parameter to be programmed is the Frequency Span. This parameter determines the maximum frequency displayed on the x-axis. It is determined by the user's response to the prompt asking for the approximate closed-loop bandwidth.

The approximate bandwidth, ω_b , is converted to Hertz by

$$f_b = \frac{\omega_b}{2\pi} \quad (46)$$

This frequency is converted to a character string and formed into the following command:

'EM FS,xxxx'=> 'Edit_Measurement_Frequency_Span, xxx'

Level 6.3 Send Command String

After the command strings have been formed, they must be sent to the proper instrument for execution. Using the high-level IEEE-488 interface software allows this to be done very easily. The subroutine IBUPU is described in Chapter 3 in detail. The interface to this subroutine is further simplified by using it in a subroutine which calls IBUPU. The new subroutine is called WRITEMSG and the format is as follows:

```
CALL WRITEMSG(device #, command string, error code)
```

where

device # is the device number of the instrument

command string is the character data to be sent

error code is a returned integer indicating error type

WRITEMSG simply counts the characters in the string and adds a command terminator character <LF> to the string. The string length is then incremented by one. This data and the device number is passed to the interface driver using the IBUPU calling sequence as described in Chapter III.

Example:

The following string is to be sent to the B/K 2032

'SR 1' ==> Performs a System Reset (Type 1)

The calling sequence would be as follows

```
CALL WRITEMSG(2, 'SR 1', ISTAT)
```

where

2 is the B/K device number

'SR 1' is the command string

ISTAT is the integer variable to receive the returned error code (1 if command was sent ok)

Level 6.4 Start Test Sequence

The actual initiation of the test sequence is slightly different for the Step and Frequency Response tests, however, these differences are transparent to the user. Each will be described in this section.

The Step test begins when the user presses the 'START' button to ready the B/K for triggering and to indicate to the VAX that the Wavetek 172B output should be turned on. The actual test cycle is started when the Wavetek sends the step pulse to the system-under-test. Since this signal is also connected to the B/K 2032 External Trigger input, the B/K initiates a single measurement cycle when the pulse is received.

The Frequency test is initiated when the user presses the 'START' key. This actually starts the measurement cycle since the B/K was put into a mode to begin measurements upon receipt of this key closure.

Summary

This chapter has taken each of the levels of DICES and implemented them as a software routine or shown that the process is implemented by the user during the course of events in solving the 'control problem'. Flowcharts have been used where necessary to simplify understanding. The final product of this chapter is a code which implements the design from Chapter III and performs a subset of the top-level system requirements derived for DICES in Chapter 2.

V. - Test

Introduction

This chapter discusses testing of software and hardware systems in a general sense, then testing of DICES, in particular, is discussed in detail. DICES testing consists of development and system testing. Development testing is performed on various modules of hardware and software when it is determined that the item is of a critical nature or is of high-risk. System testing is performed as a final test of DICES in the form of a control problem and ends at the actual control of the plant simulation with performance tests being done by the system analyzer.

Discussion of Test Strategies

According to Myers(43), software testing is the process of executing a program with the intent of finding errors. Errors are present under two circumstances: (1) if the program does not do what it is supposed to do and (2) if it does do what it is not supposed to do. Since errors in programs are inevitable, program testing is most productive when emphasis is placed on finding errors rather than attempting to demonstrate that a program is error-free. It is very difficult, if not impossible, to demonstrate the latter.

The two basic test strategies generally used are Top-Down and Bottom-Up testing.

Top-Down Testing

Top-down testing tests the high-level modules first, using test stubs to simulate the lower level modules. The stubs are 'dummy modules' that replace actual code and appear to the higher-level modules to be actual code, but are just routines to pass data in and out of the module.

The primary advantages of top-down testing are:

1. The user can see an early version of the system operation and determine if the end system will meet requirements.
2. Serious design flaws and interface incompatibilities are surfaced early in the testing.
3. Requirements for machine test time are more evenly spread out over the testing cycle.

Bottom-Up Testing

Bottom-up testing tests the lower-level modules first, and works up to the next higher-level modules after confidence is gained with the low-level code. These low-level modules are integrated into a larger module until the entire system is complete. Bottom-up testing requires test drivers to simulate the operation of the next higher level module.

The primary advantages of bottom-up testing are:

1. Frequently the only way to test a module with no
subordinates

2. Allows early test of high-risk modules

In practice, a combination of both methods is generally used.

General Test Plan

The testing on DICES will be accomplished using both top-down and bottom up testing. However, bottom-up testing will be emphasized because of the desire to obtain early insight into the high risk modules.

The high risk modules are essentially the modules used in the pole/zero pairing algorithm (PAIR.THS), the module that reads the controller parameters (RDHTF.THS), and the module that changes the coefficients in the source program (CHGCO.THS). These modules will be coded and tested individually. They will then be integrated into a larger module to form the bulk of the Filter Implementation Module.

All other modules are considered medium risk and will be coded after the overall program structure is coded.

The hardware testing will consist of verification that the Evaluation Module (containing the TMS32010) functions properly and that the input and output analog filters on the board do not cause detrimental phase shift. Various

interface tests will be conducted between the VAX, EVM, B/K Analyzer, and analog computer.

Software Test Plan

This section describes the software tests that will be performed on the three main modules.

Module: PAIR.THS

Function: Pair poles and zeros into second order sections for later implementation

Test Input: 12 combinations of poles and zeros which will test various paths through the pairing algorithm.

Module: RDHTF.THS

Function: Read the variable HTF from the ICECAP data file MEMORY.DAT.

Test Input: HTF function contained within the MEMORY.DAT data file.

Module: CHGCO.THS

Function: Enter the integer coefficients into the TMS32010 source file which will assembled into the controller object code.

Test Input: File of test coefficients and the generic controller source program 3DFILT.THS.

Hardware Test Plan

Item: Evaluation Module

Function:

1. Interface to VAX computer system
2. TMS32010 microprocessor functions as digital controller
3. Analog-to-Digital and Digital-to-Analog converters test inputs: DC voltages that cover the range of the A/D and D/A converters (± 10 volts).

System Test Plan

This section will present a simple design problem and its implementation using DICES as a system test of DICES will be presented later in this chapter. It is somewhat of a 'textbook' problem, but it is representative of a typical class of problems that can be modelled using linear, time-invariant system equations. The basic problem is taken from Phillips and Nagel (13:239-242), however, the motor inductance has not been neglected as in the reference.

Problem Statement

It is desired to design and implement a digital

controller for a servo-system (positioning system) for an antenna tracking system. In this system, an electric motor is used to rotate a radar antenna which automatically tracks an aircraft. The error signal, which is proportional to the difference between the antenna and the line-of-sight to the aircraft, is amplified and drives the motor in the appropriate direction so as to reduce the error (see fig 44). The motor used is a DC motor with armature control and a constant field current.

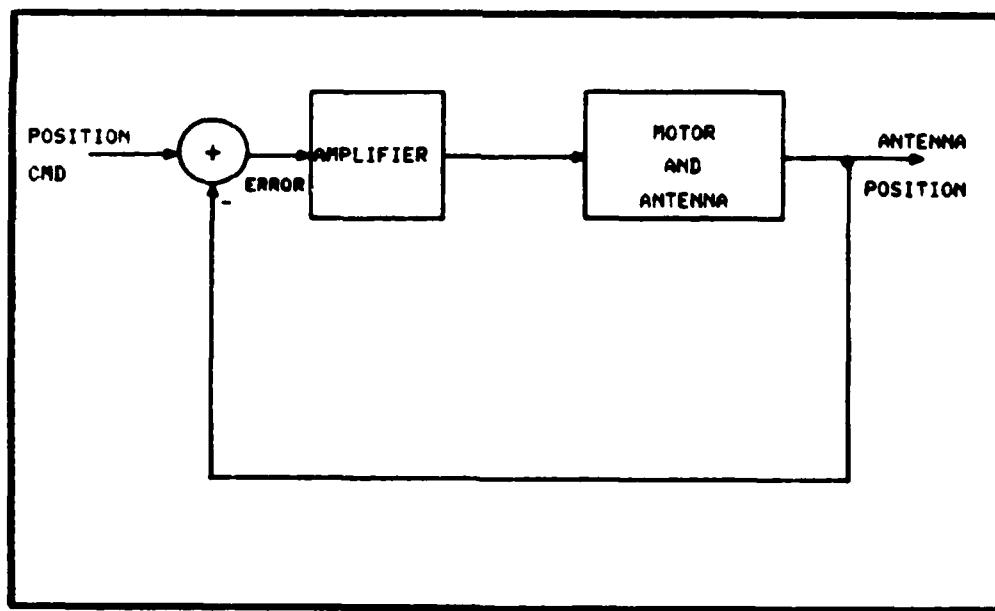


Figure 44. Servo Control System

Test Results

Software Test Results

See Appendix D for detailed test results for each of the major tests.

Hardware Test Results

It was discovered that the A/D and D/A board (referred to as the AIB) contained 50 - 4700 Hz band-pass filters on the input and the output of the board. There was also no Sample-and-Hold (S/H) circuit on the input. The input filter was used for anti-aliasing and the output filter was used for smoothing of the output signal. Both filters were sixth-order bi-quad Butterworth filters with a 50 Hz. high-pass filter preceding each one. These filters presented a problem for using the AIB as part of a digital controller for several reasons.

First, without DC coupling, the controller can not pass the very low frequency signals that exist as the controlled variable approaches steady state. This controlled variable is returned to the summer to generate an error signal. It is this error signal that contains frequencies from DC up to the highest frequency passed by the plant.

Secondly, the Butterworth filters have a large phase

variation with frequency. Since phase is an integral part of the overall control problem, it is not desired to have this sixth-order transfer function preceding the plant.

The absence of a Sample-and-Hold circuit on the input effectively allowed an aperture time equal to approximately the conversion time of the A/D converter. For the Analog Devices ADC80 Successive Approximation A/D Converter, this was 25 microseconds (34). Using this time as the approximate total aperture time (neglecting several other small delays), it can be shown (6:275) that the maximum frequency which can be sampled at rated accuracy is

$$f_{\max} = \frac{1}{(2^q)\pi\tau_a} \quad (47)$$

where q = number of bits

τ_a = aperture time of A/D convertor

f_{\max} = maximum frequency that can be accurately sampled

which yields $f_{\max} = 3.108$ Hz.

This is not sufficient for digital controller applications so it was decided to add a S/H circuit and unity gain 0 - 16 kHz amplifiers to act as buffer/isolation devices for the A/D and D/A integrated circuits. It was also decided that a S/H was not required on the output of

the D/A device because the input to it is from a latch register that is held constant during the entire sample period, T.

Filter Modifications. As stated previously, the modifications to the filters were to install unity gain, 16 kHz cut-off filters which allowed DC coupling of the input and output circuits. These unity gain amplifiers also provided isolation for the A/D and D/A integrated circuits.

The AIB contained 3 operational amplifiers for each filter section. The filter characteristics are determined by a removable header plug in which the components are mounted. The 50 Hz high-pass filter was simply an RC network at the input to the Butterworth filters. The header plug was modified to remove one of the op-amps from each of the filters. Matched resistor pairs were then mounted to form a standard single pole low-pass filter for each remaining op-amp. This yielded a double-pole, low-pass filter for the input and the output of the AIB. The frequency for these filters was chosen to be approximately 16 kHz because of the maximum sampling rate of the AIB of 40 kHz. This implied a maximum input frequency of about 20 kHz. The use of standard components produced a cut-off frequency of about 16 kHz. Appendix F describes shows the AIB board as modified for this thesis investigation.

Sample-and-Hold. Since the input to the A/D converter

contained no S/H circuit, it was decided to insert one at the output of the unity-gain amplifiers.

The output of the new low-pass filter is routed to the S/H device. This device requires a signal to put it into the 'hold' mode. This signal is provided from the A/D converter by the STATUS pin. This pin goes 'high' when the converter begins a conversion cycle. When the conversion is complete, the STATUS line goes low. This puts the S/H into the 'sample' mode which allows it to track the input signal in preparation for another HOLD operation.

The S/H was configured to function as a unity-gain S/H device with an internal hold capacitor of 1000 pf. This capacitor determines the amount of output voltage 'droop' during the hold period. This droop is not a problem for a S/H used with a fast A/D converter. This is because the voltage droop is critical only during the actual conversion time of the A/D converter, which is 25 microseconds for the ADC80.

The voltage of the S/H must stay within 1/2 LSB (.00244 volts for the AIB) during the conversion period. This implies an output voltage relation of

$$\frac{dv}{dt} < 97.65 \text{ v/s} \quad (48)$$

The dv/dt of the ADC80 is 1v/s (34), which is nearly

two orders of magnitude better than the requirements for this application. Appendix F shows the the configuration of the input circuit on the AIB.

System Test Results

This section presents the general procedures used to solve a control problem using DICES. Since the details of this procedure are described in Appendix A, DICES Users Manual, minor operational details are omitted for clarity. This example serves as the final integrated system test of DICES.

Test Procedure and Results. Note that in the following procedures, computer system responses are typed in UPPER CASE and user input is typed in lower case. The only exception to this is when a control character is typed by the user. For example , 'control T' is shown as ~T .

System Connection. The EVM must be connected as detailed in the DICES' Users Manual. The EVM must be connected to a VT-100 terminal from the rear connector labelled TERM. The EVM must also be connected to the VAX system at the connector labelled VAX. Consult the laboratory engineer or VAX system manager for assistance if required. This is the minimum system configuration to sign on to the VAX system.

VAX Sign-On

The first command given is to toggle the EVM into the TRANSPARENCY MODE. This allows access to the VAX VMS system.

DICES presents the opening menu:

DIGITAL INTERACTIVE CONTROLLER EVALUATION SYSTEM (DICES)

OPTIONS:

1. SYSTEM PERFORMANCE SPECS
2. MATHEMATICAL MODEL DEVELOPMENT
3. PLANT SIMULATION
4. CONTROLLER DESIGN
5. IMPLEMENT CONTROLLER
6. CLOSED-LOOP PERFORMANCE TESTING
7. REPORT GENERATION
8. END PROGRAM

Step 1. DETERMINE SYSTEM PERFORMANCE SPECIFICATIONS

Selecting option 1, the user is directed to determine the system performance specifications of the system under study. This step is performed by the user prior to using DICES. The user/designer analyzes the requirements of the system and quantifies them using standard control system

figures of merit.

For this example, the system performance requirements are as follows (for a unit step input):

- < 20% peak overshoot (M_p)
- < 7.5 seconds peak time (t_p)
- < 15 seconds settling time (t_s)

The main menu is again presented to the user.

Step 2. DEVELOP PLANT MODEL

Option 2 is selected which directs the user to develop the mathematical model of the plant to be controlled. This step is performed external to DICES. The mathematical model for the plant (in this case the amplifier and motor) is developed and used as the basis for the design of the controller.

The servomotor model is shown in figure 45. It is now necessary to determine the transfer function of the motor model.

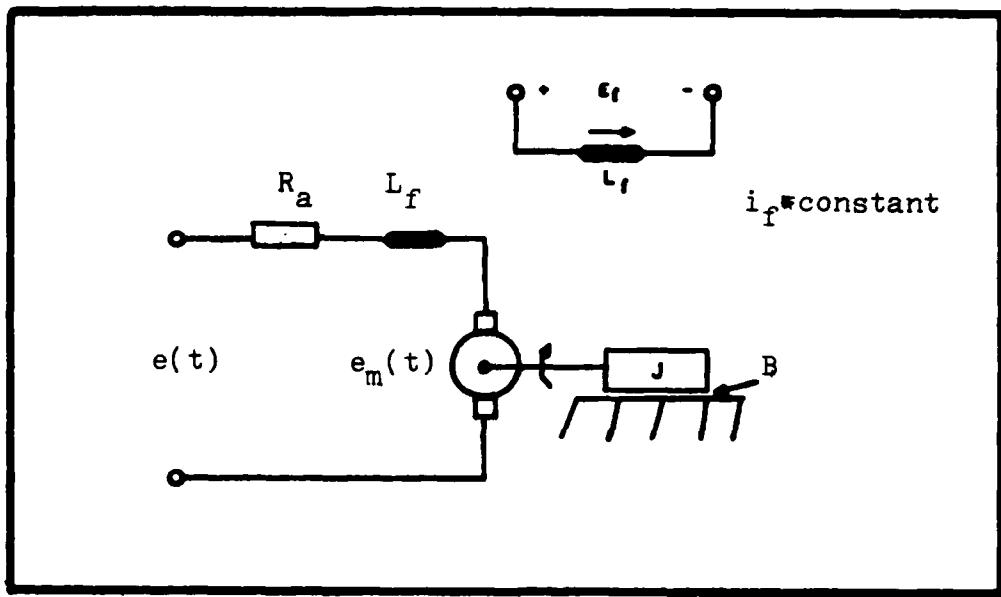


Figure 45. Servomotor System

To determine the transfer function of the motor, the differential equations that describe the system dynamics are written. Appendix A derives the servomotor transfer function in detail. Note that initial conditions are neglected since a transfer function is being obtained.

The plant transfer function is

$$\frac{\delta(s)}{E(s)} = \frac{K_T/JL_a}{s[s^2 + \left(\frac{BL_a + R_aJ}{JL_a}s \right) + (R_aB + K_TK_b)]} \quad (49)$$

where

θ is the motor shaft output position

E is the input command signal

B is the viscous damping of the motor

J is the motor inertia

R_a is the armature resistance

L_a is the armature inductance

K_T is a motor constant

K_b is a motor constant

So it can be seen that the plant model is a third-order transfer function. Had the armature inductance, L_a been neglected (as is often done), the plant model would have been second-order.

After inserting typical values for the various parameters, the plant transfer function is obtained.

$$G_p(s) = \frac{2}{s(s + 1)(s + 2)} \quad (50)$$

Step 3. SIMULATE PLANT MODEL

The transfer function for the plant may be simulated on an analog computer as described in Appendix A (see

figure 46).

After wiring the above plant model and closing the feedback loop, it can be seen that the actual simulation results closely follow that of the uncompensated theoretical results (see figures 47 and 48).

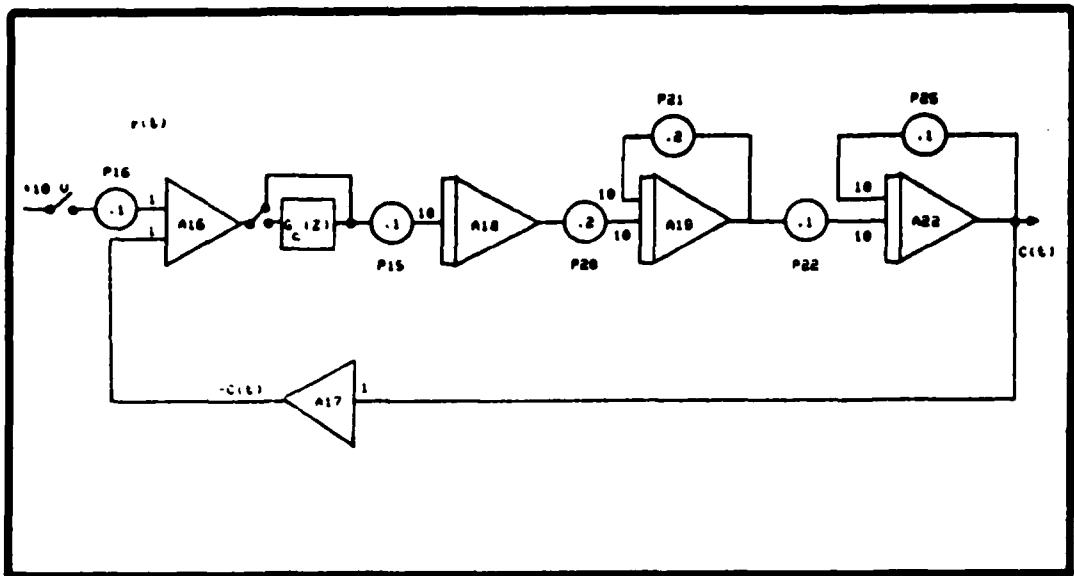


Figure 46. Analog Computer Simulation

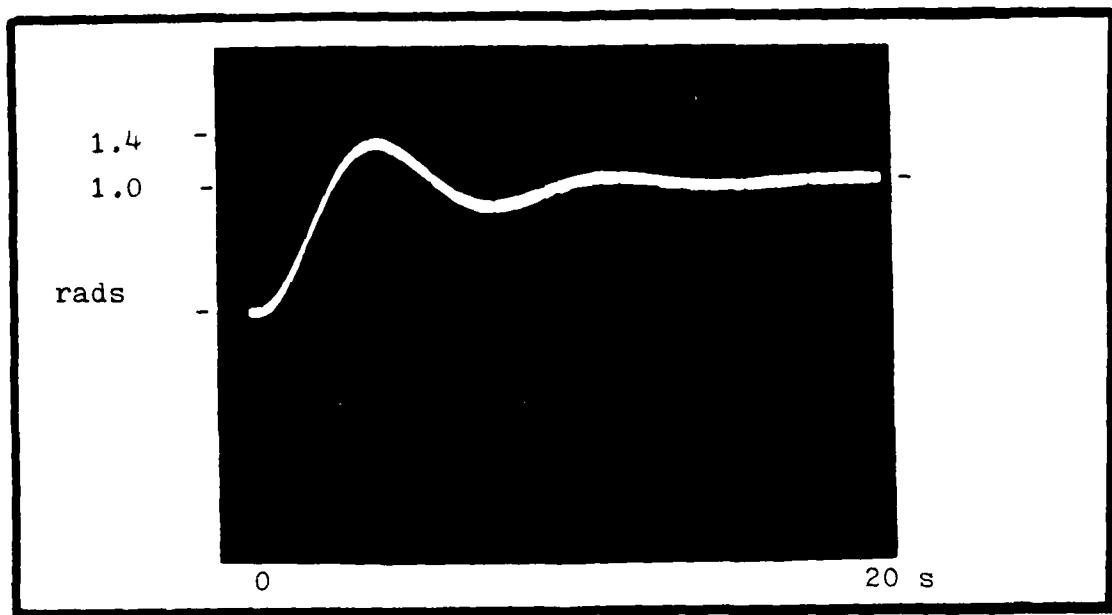


Figure 47. Uncompensated Actual Closed-Loop Simulation

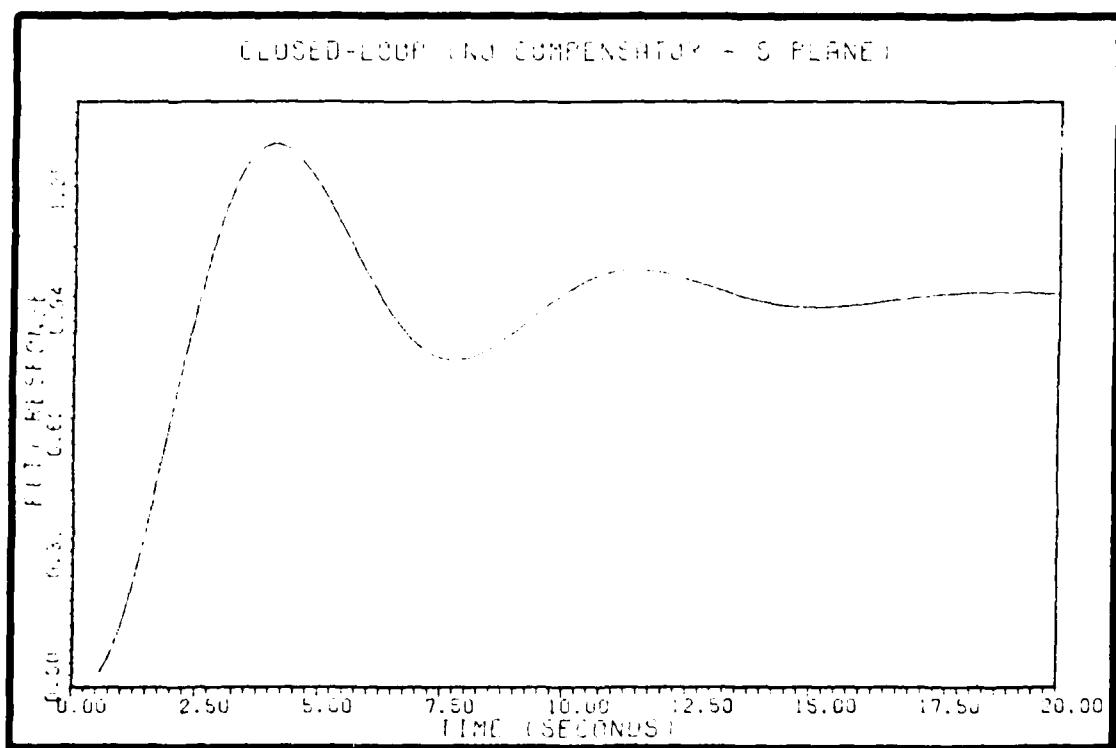


Figure 48. Uncompensated Theoretical Closed-Loop Simulation

ZOH and Sample-Period Effects. After the plant was tested with no sampler or ZOH, various tests were performed that examined the effects of inserting a ZOH and sampler into the forward loop of the control system. The sample-period was varied to see the effects of lengthening this parameter. Complete details of this test are given in Appendix A as part of the example problem.

The step response for the various sample rates are given below :

T_s	t_p	t_s	M_p	Comments
.05	4.0	16	1.4	Basic System
.05	3.8	17	1.4	Sample/Hold Only
.26	3.9	20	1.45	Sample/Hold Only
.65	4.3	>20	1.5	Sample/Hold Only
1.31	5.35	>50	1.8	Sample/Hold Only

Table 18. Theoretical Results

T_s	t_p	t_s	M_p	Comments
.05	3.8	17	1.4	Basic System
.05	3.8	17	1.4	Sample/Hold Only
.26	3.9	19	1.45	Sample/Hold Only
.65	4.3	>20	1.5	Sample/Hold Only
1.31	5.35	>50	1.8	Sample/Hold Only

Table 19. Test Results

where

T_s is Sample-Period in seconds
 t_p is peak time in seconds
 t_s is settling time in seconds
 M_p is peak value in volts

It can be seen that there is little effect from a sample period of .05 seconds or .26 seconds. However, when the sample-period is .65 or 1.31 seconds, the stability of the system becomes much worse.

Step 4. CONTROLLER DESIGN

Option 4 from the main DICES menu is selected which transfers the user to ICECAP. From here the controller is designed to meet the system performance requirements. The details of the actual controller design are omitted for brevity. However, the plant digitization is shown. It is also noted that gain alone will not adequately correct the performance problems. A frequency response method of compensator design is used in order to produce a phase margin of 55 degrees.

The sample period is taken to as .05 seconds, which is one-tenth the fastest time-constant in the plant. The Z-transfer function of the plant transfer function is taken using the impulse invariance technique. A Zero-Order-Hold (ZOH) is correctly assumed to be present at the output of the EVM controller. This yields

$$z[G_p(z)] = \frac{z - 1}{z} z \left[\frac{2}{s^2(s + 1)(s + 2)} \right] \quad (51)$$

$$Z[G_p(s)] = \frac{.00004014(Z + 3.595)(Z + .25)}{(Z - 1)(Z - .9512)(Z - .9048)} \quad (52)$$

The loop was first closed with only a sampler and ZOH to determine if there are negative effects due to the sample-and-hold process. It can be seen from the last section that the ZOH alone with a sample period of .05 seconds had no negative impact on the system response. Thus, the .05 second sample-period is a very conservative selection.

Now, using the plant Z-transfer function as the basis for the compensator design yields the following:

$$G_c(s) = \frac{.3974(Z - .995)}{(Z - .998)} \quad (53)$$

Forming the closed-loop system using ICECAP yields the following analytical step-response characteristics:

$$M_p = 1.2$$

$$t_p = 7 \text{ seconds}$$

$$t_s = 14 \text{ seconds}$$

$$\delta_{ss}(t) = 1 \text{ radian}$$

which meet the requirements originally imposed on the system.

Upon completion of this step, the user enters the ICECAP command STOP which exits the ICECAP program and saves the design parameters for implementation.

Step 5. FILTER IMPLEMENTATION

The user is presented with the main DICES menu following the exiting of ICECAP. The user selects option 5 which is the Filter Implementation Menu.

CONTROLLER IMPLEMENTATION

OPTION:

- 1. READ DESIGN PARAMETERS**
- 2. ROUND-OFF OR TRUNCATION**
- 3. FORM SECOND-ORDER SECTIONS**
- 4. GENERATE OBJECT CODE**
- 5. DOWNLOAD OBJECT CODE**
- 6. RETURN TO MAIN MENU**

Each of the steps required to implement the controller will now be presented.

The first function executed is to read the design

parameters and pair the poles and zeros into second-order sections. Selecting option 1 of the Implementation Menu produces the simple pairing of the pole and zero, since this is a first-order compensator.

```
DICES  
3D DIGITAL CONTROLLER IMPLEMENTATION  
*****  
NUMBER OF  
SECOND-ORDER SECTIONS      1  
NUMERATOR ORDER            1  
DENOMINATOR ORDER          1  
LOOP SENSITIVITY           .3479000  
*****  
UNPAIRED POLES AND ZEROS  
*****  
CZERO  
(0.9950000,0.0000000E+00)  
  
CPOLE  
(0.9980000,0.0000000E+00)  
*****  
SECOND-ORDER SECTION PAIRINGS  
*****  
SECTION NUMBER =             1  
*****  
CORDPZ ZEROS(1) = (0.9950000,0.0000000E+00)  
CORDPZ ZEROS(2) = (9999.000,9999.000)  
CORDPZ POLES(1) = (0.998000,0.0000000E+00)  
CORDPZ POLES(2) = (9999.000,9999.000)
```

Figure 49. Pole/Zero Pairing

Option 2 is then selected which allows selection of round-off or truncation of the integers which are formed during the quantization process as discussed in the design

chapter. Round-off will be selected for this example problem.

Option 3 forms the second-order sections with actual coefficients and sample period integers that will be assembled into the TMS32010 object code. The coefficients and sample period integers formed as a result of this step are shown below.

```
QUANTIZATION ROUTINES
=====
NUMBER OF SECTIONS           1
HK = 0.3973999990033917
TSAMP = 5.000001E-02
=====
SECTION NUMBER               1
=====
B0 = 0.3974363820599411
B1 = -0.3954492020447657
IB0 =      13023
IB1 =     -12957
A1 = 0.9980000257492065
IA1 =      32702
=====
TSAMP = 5.0000001E-02
ITSAMP =      419
```

Figure 50. Second-order Section Data

Option 4 assembles the source program into executable object code for the TMS32010. Upon selection of this option, a prompt will appear asking for the filename of the

source code. The response must be 3DFILT.THS. After this entry, the user simply hits the RETURN key to enter the remaining default file names. Upon completion of the assembly process, the user is returned to the Implementation menu. The file 3DFILT.MPO contains the object code in TMS9900 format which permits loading by the Evaluation Module.

Option 5 is now selected to load the object code into the Evaluation Module for later execution by the TMS32010.

Step 6. PERFORMANCE EVALUATION

To evaluate the performance of the digital controller in a closed-loop system, option 6 is executed.

The current version of DICES can obtain the step response and frequency response of the closed-loop system. Each of these tests will be discussed in the following sections.

Step Response. From the Performance Evaluation menu, select option 1 - Step Response Test. DICES will prompt for AMPLITUDE OF the STEP INPUT. Enter a carriage return for this value, since DICES defaults to a 1.0 volt input step. The next input is the APPROXIMATE SETTLING TIME for which 17 seconds is entered.

The test is started and after the test is complete, the B/K 2032 is placed into a mode that allows the user to move the cursor along the step response to determine the system peak time, settling time, peak value, etc. The value is read from the B/K 2032 screen in the upper right-hand corner of the screen.

With the compensator in the closed-loop, the analytical step response is shown in figure 51.

The figures of merit are:

$$t_p = 7 \text{ seconds}$$

$$M_p = 1.2 \text{ units}$$

$$t_s = 14 \text{ seconds}$$

The actual results of the test performed during this example are shown in figure 52. The figures of merit are

$$t_p = 6.5 \text{ seconds}$$

$$M_p = 1.2 \text{ units}$$

$$t_s = 11 \text{ seconds}$$

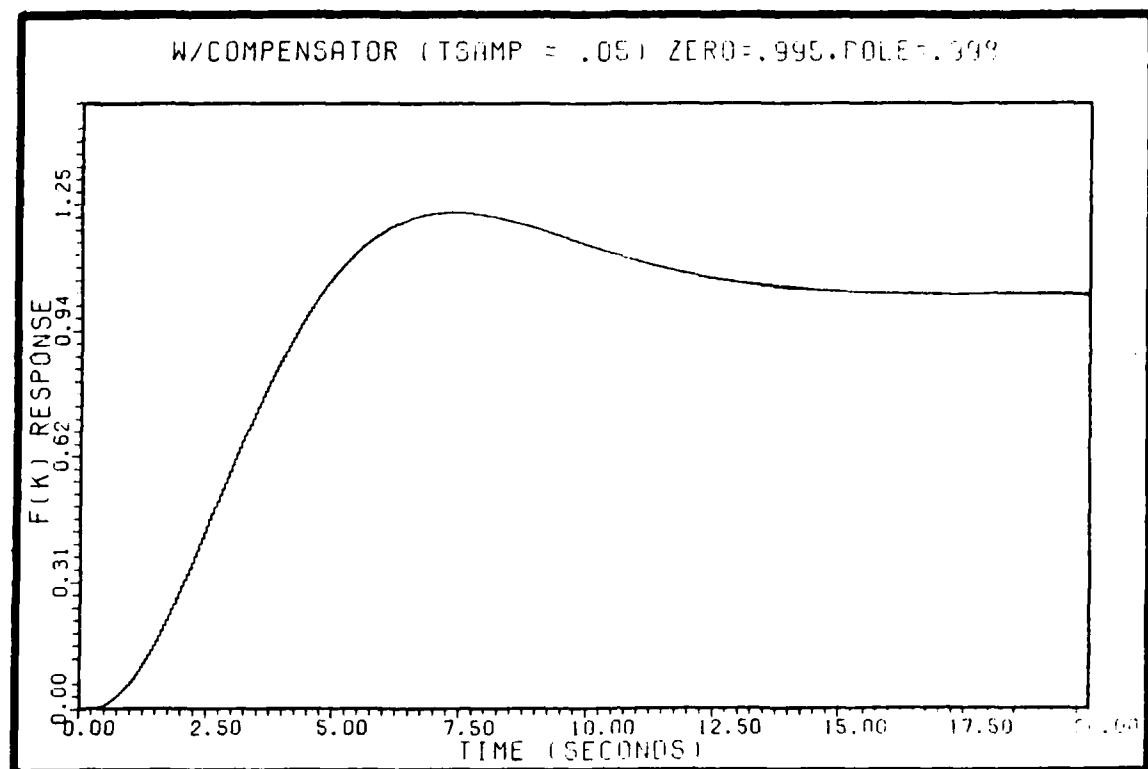


Figure 51. Compensated Theoretical Step Response

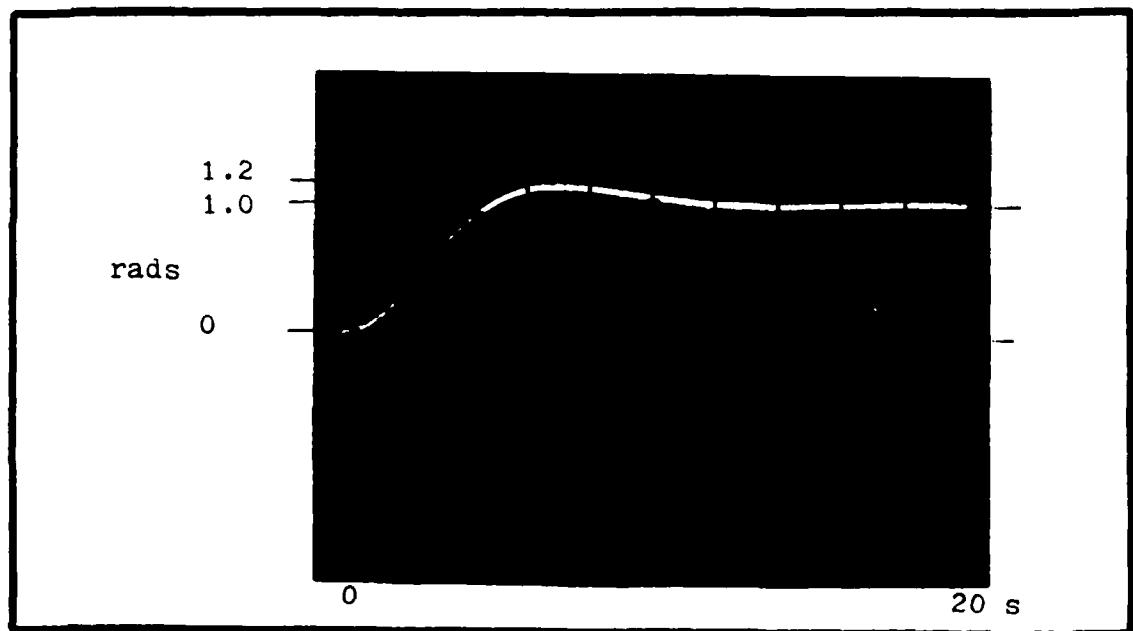


Figure 52. Actual Compensated Step Response

As can be seen, the performance with an actual 16-bit digital filter implementation is nearly the same as the theoretical results obtained with ICECAP.

Frequency Response. From the Performance Evaluation menu, select option 2 - FREQUENCY RESPONSE TEST. DICES will prompt for APPROXIMATE BANDWIDTH OF CLOSED-LOOP SYSTEM. Enter 10 for this value.

The test will take approximately 17 MINUTES because of the low bandwidth of the system. This long test time is a consequence of the B/K method of performing the frequency response test of a system. After the test is complete, the B/K 2032 is placed into a mode that allows the user to move the cursor along the magnitude and phase curves simultaneously. The magnitude (in dB) and the phase of the plant output (in degrees) can be read from the right-hand top and bottom of the screen, respectively.

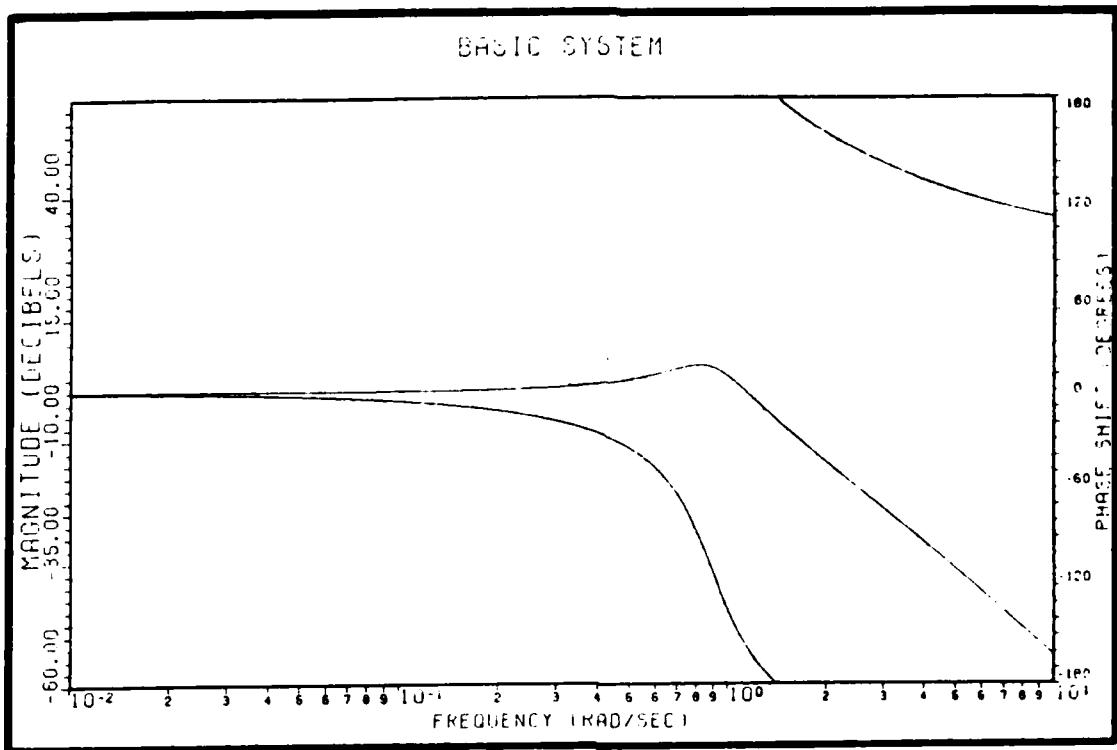


Figure 53. Theoretical Uncompensated Frequency Response (Basic System)

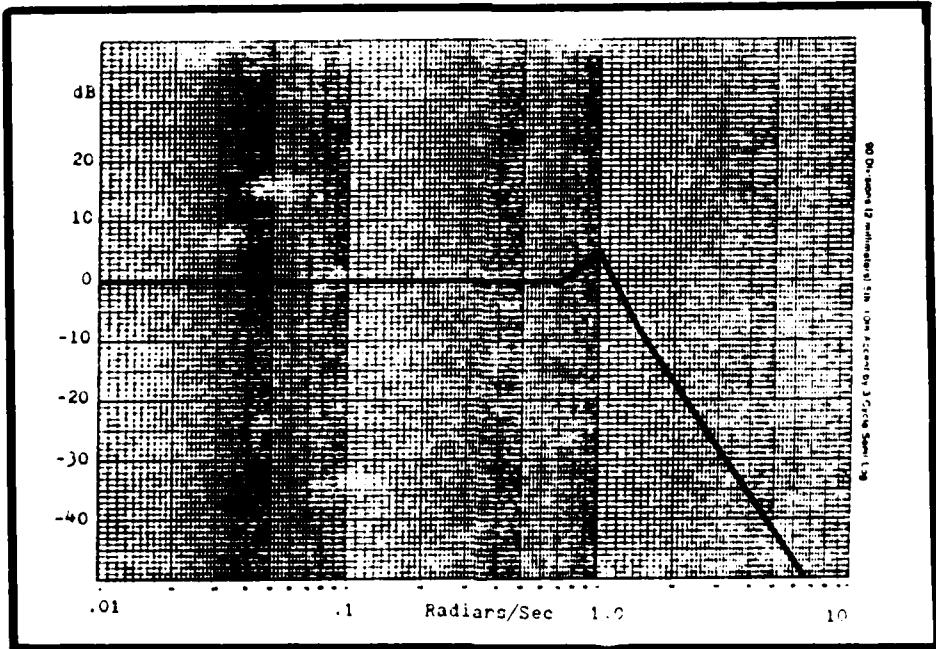


Figure 54. Actual Frequency Response of Uncompensated System (Basic System)

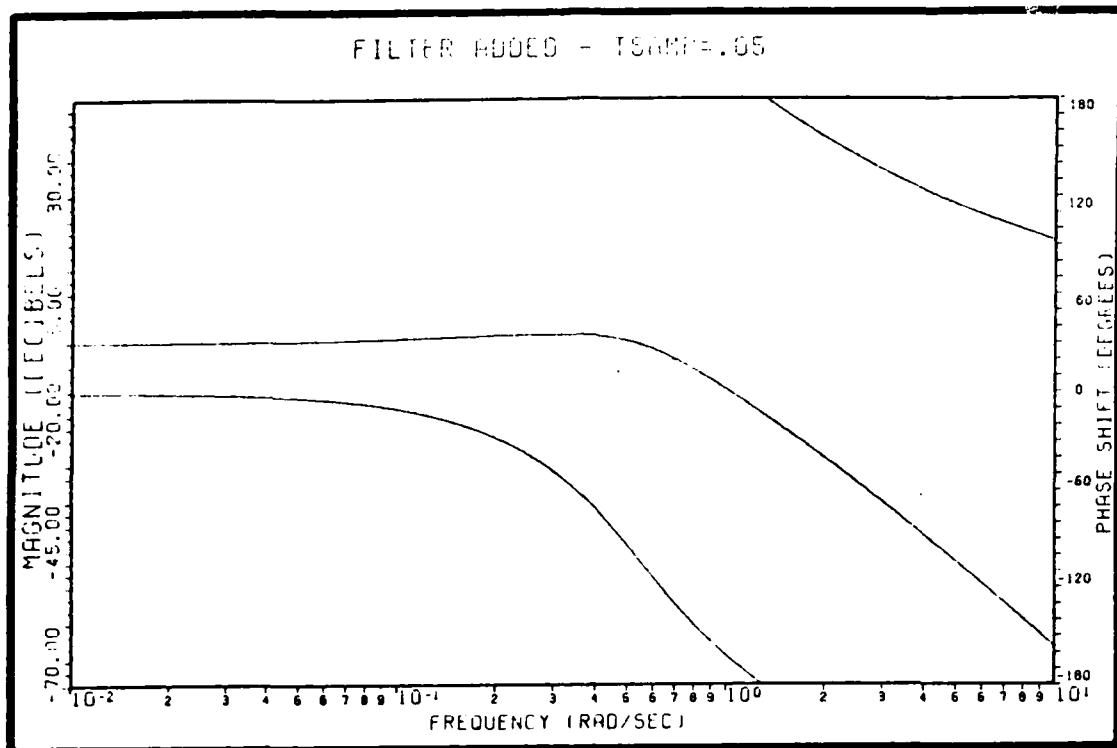


Figure 55. Theoretical Frequency Response
of Compensated System ($T_S = .05$)

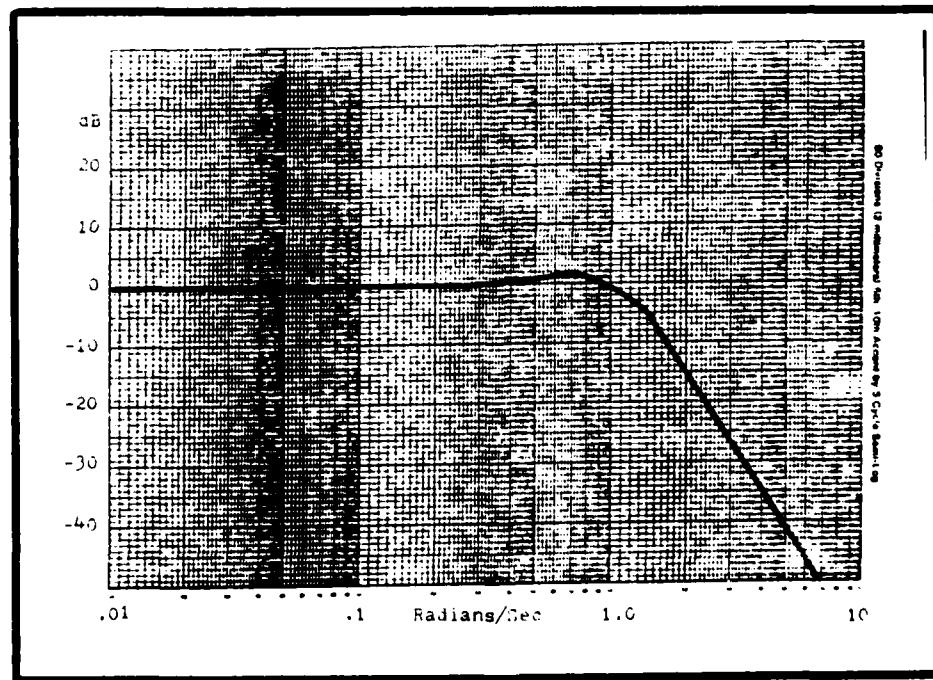


Figure 56. Actual Frequency Response
of Compensated System ($T_S = .05$)

As can be seen from the frequency response data, the bandwidths of the uncompensated and compensated systems are:

	ω_m	ω_b	M_m
Basic	.9	1.3	1.77
Compensated ($T_S = .05$)	.5	.7	1.15

Table 20. Frequency Response Theoretical Results

	ω_m	ω_b	M_m
Basic	1.0	1.2	1.7
Compensated ($T_S = .05$)	.6	.7	1.25

Table 21. Frequency Response Test Results

Summary

This section has shown how a typical control problem is taken from beginning to end using DICES. The user is free to change filter designs (within the current limitations) and make changes to the plant model if so desired. Each of the tests can be executed again if so desired.

Chapter VI - Conclusions and Recommendations

Conclusions

This investigation, Development of a Digital Interactive Controller Evaluation System (DICES), was carried out in a typical hardware or software system development cycle. The requirements were analyzed and specified early in the investigation. These requirements were then used to design the system. Implementation of the design took place in chapter IV with both hardware and software contributing to the final system. Testing concepts and results were addressed after the final system implementation was complete.

This investigation resulted in the establishment of an interactive system which permits implementation of a digital controller design which is to compensate a known plant. The controller is implemented on a TMS32010 Digital Signal Processor whose instruction set is optimized for DSP applications. The user simulates the plant using an analog computer and then connects the controller into the system to form a closed-loop control system. Testing in the form of step response and frequency/phase response is accomplished using test equipment programmed via the IEEE-488 interface bus under control of a VAX 11/780. The results of the tests are available graphically on the system analyzer and can be retained after the test for

further analysis.

A complete example problem is used as the final system test. This example validates the performance of DICES. Analytical results are obtained and 'favorably' compared with actual results using control systems CAD packages such as ICECAP and TOTAL.

Recommendations for Further Investigation

Recommendations fall generally into the areas of continued DICES software and hardware development and other recommendations.

Software

1. Development of a remote interrupt capability by the VAX 11/780 of the TMS32010. This would alleviate the need to manually RESET the TMS32010 to stop the filter program at the end of each test. A modification of the TMS32010 code to allow monitoring of an interrupt pin is a possible approach to this recommendation.

2. Development of a more general scaling routine for the generic 3D (13) digital compensator which would avoid the current restriction of filter gains of less than one. Current system must incorporate additional gain into the analog computer.

3. Add additional filter types (1D, 2D, etc) to implementation options (6,13), i.e., a library of TMS32010 programs that implement various filter structures.

4. Development of higher-order compensator capability. The current limitation is an eighth-order over eighth-order digital compensator.

5. Implement a more extensive and general interface with the VAX 11/780 system for transferring data from the IEEE-488 instrumentation to the VAX system.

Hardware

1. Development of an input/output analog multiplexer to allow the use of DICES in a Multiple-Input-Multiple-Output (MIMO) control system. The MIMO problem is a more realistic class of problems.

2. Extend the interface to the Simstar digital/analog system.

3. Add hardcopy capability to the B/K 2032 using an x-y plotter or the VAX laser printer.

Other Recommendations

It is further recommended that the development of a general-purpose IEEE-488 controlled test/data acquisition station be developed using DICES as the core. The current

laboratory inventory of IEEE-488 compatible equipment includes:

- 1) B/K 2032 System Analyzer
- 2) Wavetek 172B Signal Generator
- 3) Hewlett-Packard 1980B 100 MHz Measurement System
- 4) X-Y Plotters

Possible additional instruments that could be added to the system include:

- 1) IEEE-488 compatible Spectrum Analyzer
- 2) IEEE-488 compatible Digital Voltmeter
- 3) IEEE-488 compatible Frequency Counter

This system would allow all current DICES capabilities plus extended bandwidth frequency response measurements. It would serve as an excellent teaching/demonstration tool for classes studying topics in the areas of control systems design/analysis and communication systems (modulation types and frequency stability). It could also serve as a general-purpose data acquisition system which would reduce the time required to configure/reconfigure a data collection system for each new laboratory investigation.

BIBLIOGRAPHY

1. Clune, Thomas R. 'Interfacing for Data Acquisition,' BYTE, 10: 269-282 (February 1985).
2. TMS32010 Evaluation Module User's Guide, SPRU005. Texas Instruments, Incorporated, February 1984.
3. TMS32010 Assembly Language Programmer's Guide, SPRU002B. Texas Instruments, Incorporated, November, 1983.
4. Implementation of FIR/IIR Filters with the TMS32010, SPRA003. Digital Signal Processing Application Report. Texas Instruments, Incorporated, August 1984.
5. D'Azzo, John J. and Constantine H. Houpis. Linear Control Systems Analysis and Design. (Second Edition). New York: McGraw-Hill Book Company, 1981.
6. Houpis, Constantine H. and Gary B. Lamont. Digital Control Systems. New York: McGraw-Hill Book Company, 1985.
7. Gilbert, Richard. 'The General Purpose Interface Bus,' IEEE Micro, 2: 41-51 (February 1982).
8. Nagle, Jr. H. T. and V. P. Nelson. 'Digital Filter Implementation on 16-Bit Microcomputers,' IEEE Micro, 1: 23-41 (February 1981).
9. Follett, Randolph F., Jerrel R. Mitchell, and L. L. Gresham. 'A Computer Code for Emulation of Microprocessor Arithmetic Errors,' SYST 83: 98-101.
10. Integrated Circuits. Volume 1. Analog Devices, Inc. 1984.
11. Rattan, Kuldip S., and Alok Sarwal. 'Real-Time Analysis of a Digital Multiloop Flight Control System,' NAECON84, 1: 534-539 (1984).

12. Maybeck, Peter S.. Stochastic Models, Estimation, and Control.1. Orlando: Academic Press, Inc, 1979.
13. Phillips, Charles H., and H. Troy Nagle, Jr. Digital Control System Analysis and Design. Englewood Cliffs, NJ: Prentice Hall, Inc, 1984.
14. Peled, Abraham, and Bede Liu. Digital Signal Processing, Theory, Design, and Implementation. New York: John Wiley and Sons, 1976.
15. Larimer, Stanley J. An Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS Thesis. Wright Patterson AFB, OH: Air Force Institute of Technology, March 1978.
16. Logan, Glen T. Development of an Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS Thesis. Wright Patterson AFB, OH: Air Force Institute of Technology, March 1982.
17. Tausworthe, Robert C. Standardized Development of Computer Software. Englewood Cliffs, New Jersey: Prentice-Hall, 1979.
18. Meadow, Charles T. Man-Machine Communications. New York, New York: John Wiley and Sons, Inc, 1970.
19. Weinberg, Victor. Structured Analysis. New York, New York: Yourdon Press, 1979.
20. Thames, J. Wayne. ALR-46 Computer Graphics System, MS Thesis, Wright-Patterson AFB, OH: Air Force Institute of Technology, December 1981.
21. Kernighan, B. W. and P. J. Plaugher. The Elements of Programming Style. New York: McGraw-Hill Book Company, 1974.
22. Bauer, F. L. (Editor). Software Engineering: An Advanced Course. New York: Springer-Verlag, 1977.

23. Grogono, Peter. Programming in PASCAL. Revised Edition. Reading, MA: Addison-Wesley Publishing Company, Incorporated, 1980.
24. Jackson, L. B. 'Roundoff Noise Analysis for Fixed Point Digital Filters Realized in Cascade or Parallel Form,' IEEE Transactions on Audio Electroacoustics, AU-18: 107-122, June, 1970.
25. Kan, E. P. F., and J. K. Aggarwal. 'Minimum-Deadband Design of Digital Filters,' IEEE Transactions on Audio Electroacoustics, AU-19: 292-296, December, 1971.
26. Maria, G. A., and M. F. Fahmy. 'Limit Cycle Oscillations in a Cascade of First and Second Order Digital Sections,' IEEE Transactions on Circuits and Systems, CAS-22, 131-134, February, 1975.
27. Steiglitz K., and B. Liu. 'An Improved Algorithm for Ordering Poles and Zeros of Fixed-Point Recursive Digital Filters,' IEEE Transactions on Acoustics and Speech Signal Processing, ASSP-24: 341-343, August, 1976.
28. Carlson, Alan, George Harnauer, Thomas Carey, and Peter J. Holsberg (Editors). Handbook of Analog Computation. (Second Edition). Princeton, New Jersey: Electronic Associates, Inc, 1967.
29. Logan, Glen T. Development of an Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. ICECAP User's Manual. MS Thesis. Wright Patterson AFB, OH: Air Force Institute of Technology, March 1982.
30. TMS32010 Analog Interface Board Data Manual. (Release 2). Texas Instruments, Incorporated, 1983.
31. IEEE-488 Standard Digital Interface for Programmable Instruments. IEEE, New York, 1978.
32. GPIB11-2 Operating and Service Manual. National Instruments Corporation, Austin, Texas, July, 1982.

33. GPIB11 Series Multiboard Driver VAX VMS Software Reference Manual. P/N 310005-04. National Instruments Corporation, Austin, Texas, March, 1985.
34. Analog-to-Digital Convertors, Volume 1, 10-83. Specification Sheet for ADC80. Analog Devices.
35. Ricci, Fred J. Analor/Logic Computer Programming and Simulation. New York: Spartan Books, 1972.
36. Instruction Manual for the Wavetek 172B Programmable Signal Source. Wavetek, Incorporated, October, 1979.
37. B/K 2032 Instruction Manual IEC/IEEE Interface.3. Brueel and Kjaer. September, 1983.
38. Gilbert, Richard. 'The General Purpose Interface Bus,' IEEE Micro.2: 41-51, February, 1982.
39. Clune, Thomas R. 'Interfacing for Data Acquisition,' Byte Magazine.10: 269-282, February, 1985.
40. B/K 2032 Instruction Manual - Operation.2. Brueel and Kjaer. September, 1983.
41. Beach, R., M. Lih, and W. Fabens. 'A Generalized Laboratory Automation System,' DECUS84: 177-182, Cincinnati, OH, June, 1984.
42. MATRIXX - Users Manual (Version 5.0). Integrated Systems, Incorporated, Palo Alto, California, January, 1985.
43. Myers, Glenford J. The Art of Software Testing. New York: John Wiley and Sons, 1979.

Appendix A - DICES Users Manual

**DIGITAL INTERACTIVE CONTROLLER
EVALUATION SYSTEM
(DICES)
USERS MANUAL**

CONTENTS

1.0 INTRODUCTION	1
2.0 WHO SHOULD USE DICES?	1
3.0 OVERVIEW	1
4.0 SYSTEM CONFIGURATION	2
4.1 HARDWARE REQUIREMENTS	4
4.2 MINIMUM SYSTEM TO COMMUNICATE WITH DICES	4
4.3 MINIMUM SYSTEM FOR IMPLEMENTATION AND TEST OF CONTROLLERS	7
4.3.1 EVM ANALOG CONNECTIONS	8
4.3.1.1 CASCADE CONTROLLER	8
4.3.1.2 FEEDBACK CONTROLLER	8
4.3.2 OTHER EQUIPMENT	11
4.3.2.1 B/K DETAILED CONNECTIONS	13
4.3.3 IEEE-488 INTERFACE BUS	19
4.3.3.1 B/K AND WAVE TEK CONNECTIONS ...	20
4.3.4 POWER CONNECTIONS	21
5.0 FUNCTION DESCRIPTIONS	22
5.1 SYSTEM PERFORMANCE SPECIFICATIONS	22
5.2 MATHEMATICAL MODEL DEVELOPMENT	22
5.3 PLANT SIMULATION	22

5.4	CONTROLLER DESIGN	23
5.4.1	SAVING CONTROLLER DESIGN	23
5.4.2	LEAVING ICECAP	23
5.5	IMPLEMENT CONTROLLER	24
5.5.1	READ DESIGN PARAMETERS	24
5.5.2	SELECT ROUNDING OR TRUNCATION	25
5.5.3	PAIR POLES AND ZEROS	25
5.5.4	GENERATE TMS32010 OBJECT CODE	25
5.5.5	DOWNLOAD OBJECT CODE	26
5.6	CONTROLLER PERFORMANCE EVALUATION	27
5.6.1	STEP RESPONSE	28
5.6.2	FREQUENCY RESPONSE	30
5.6.3	IMPULSE RESPONSE	30
5.7	REPORT GENERATION	30
5.7.1	FILTER PAIRS AND QUANTIZED COEFFICIENTS	31
6.0	EXAMPLE PROBLEM	31
6.1	DETERMINE SYSTEM SPECIFICATIONS	35
6.2	DEVELOP PLANT MODEL	35
6.3	SIMULATE PLANT MODEL	40
6.3.1	ZOH AND SAMPLE PERIOD EFFECTS	43
6.3.2	TEST CASES	44
6.4	CONTROLLER DESIGN	51
6.5	FILTER IMPLEMENTATION	53
6.6	PERFORMANCE EVALUATION	58
6.6.1	STEP RESPONSE	58

6.6.2 FREQUENCY RESPONSE	62
6.7 REPORT GENERATION	63
6.8 SUMMARY	63
7.0 KNOWN BUGS, ERRORS, AND TRICKS	63
8.0 REFERENCES	67
ATTACHMENT 1 EVM DESCRIPTION	70

DICES USER's MANUAL

1.0 INTRODUCTION

This appendix first presents an overview of the Digital Interactive Controller Evaluation System (DICES). Second, it provides an introduction to all the DICES features. Finally, it includes examples of the use of DICES to implement digital controller designs and test their performance in a closed-loop system.

2.0 WHO SHOULD USE DICES?

Since DICES draws upon all aspects of control system design and analysis, a potential user should have at least a first-course in control system analysis and design, or the equivalent knowledge or experience.

3.0 OVERVIEW OF DICES

DICES is a hardware and software system that permits interactive implementation of a digital controller that has been designed using a control systems Computer-Aided-Design package called ICECAP. This controller can then be inserted into a closed-loop control system where the plant is simulated using an analog computer. Performance tests are then conducted to determine if the 16-bit implementation of the digital controller meets the performance specifications set for the closed-loop system. The current version of DICES permits implementation of up to an eighth-order digital

controller. DICES is designed to allow easy implementation of the controller design that evolves from a design session with ICECAP. DICES assumes some knowledge of the use of the VAX VMS system and the TR-48 or equivalent analog computer system. Knowledge in the use of a B/K 2032 System Analyzer is helpful but not required to use DICES.

DICES does not assist the user in the design of a digital controller for controlling a plant. DICES does, however, allow access to ICECAP, which gives the user freedom to design a suitable controller to meet the specifications set forth for the plant under consideration.

4.0 SYSTEM CONFIGURATION

This section first discusses the minimum system necessary to communicate with DICES followed by the minimum system necessary to implement and test digital controllers. It then takes the user through step-by-step procedures for connecting the required components. Figure 1 shows the DICES block diagram.

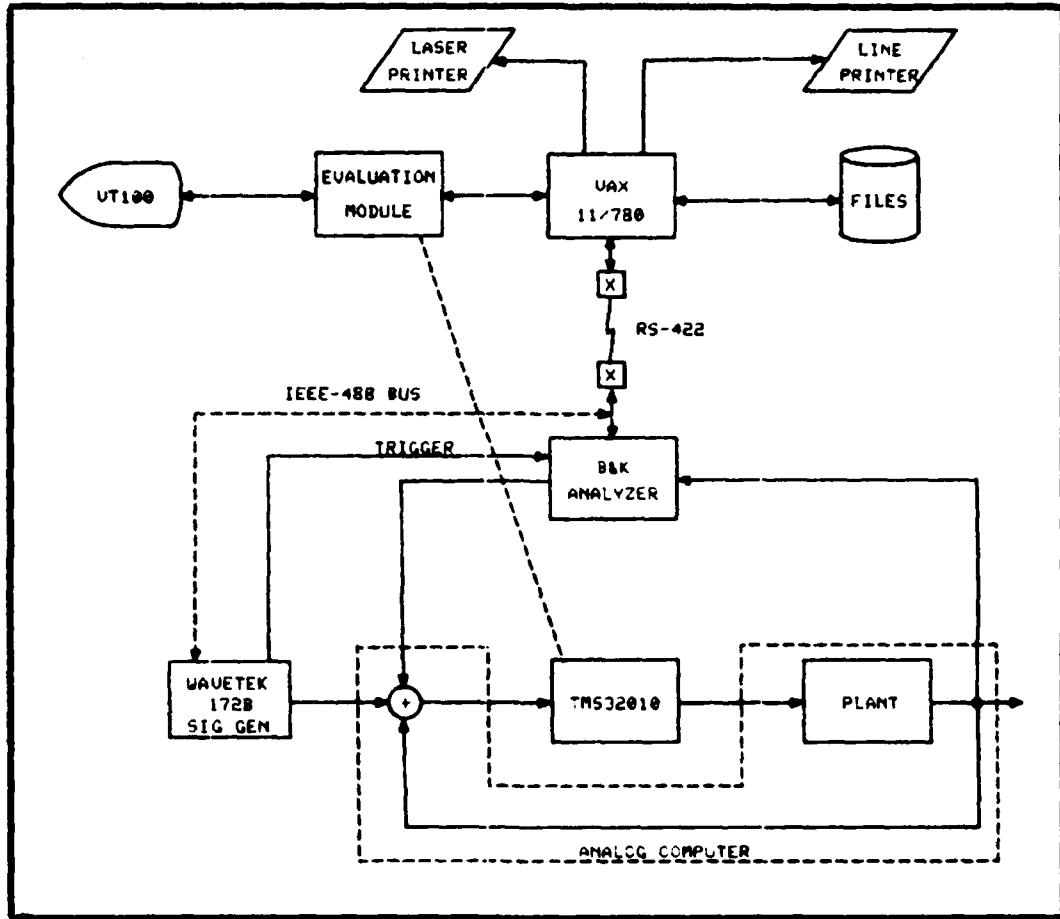


Figure 1. DICES Block Diagram

4.1 HARDWARE REQUIREMENTS

- * 1) TMS32010 Unit
- * 2) VT-100 Video Terminal or equivalent
- 3) Brueel and Kjaer (B/K) 2032 Signal Analyzer
- 4) Electronic Associates, Inc. (EAI) TR-48 Analog Computer or equivalent
- * 5) VAX 11/780 VMS system
- 6) Wavetek 172B Programmable Signal Source

* denotes minimum required to communicate with DICES

4.2 MINIMUM SYSTEM TO COMMUNICATE WITH DICES

The first step is to connect a VT-100 video terminal to the TMS32010 Evaluation Module (EVM). The VT-100 is connected to the rear terminal of the EVM labelled 'TERM'. The other connector on the rear of the EVM labelled VAX goes to a VAX VMS 11/780 remote terminal line (see figure 2). Upon completion of these connections, power is applied and the RESET switch on the EVM is activated. RETURN is pressed on the VT-100 terminal to get the EVM's 'attention', which then displays the EVM system prompt of '?' on the VT-100. The user now types the following to initialize the EVM system:

? <u>xon<cr></u>	...permits XON/XOFF protocol with the VAX
? <u>init<cr></u>	
INTERNAL <u>ext<cr></u>	...uses clock external to Analog Board in EVM
? <u>comm<cr></u>	
CNTRL C <u>ctrl t<cr></u>	...changes 'transparency mode' toggle to ~T because VAX VMS uses ~C
Note: ~T = 'ctrl t'	

Note: Response from EVM is in UPPER case.

User-supplied data is lower-case and underlined.

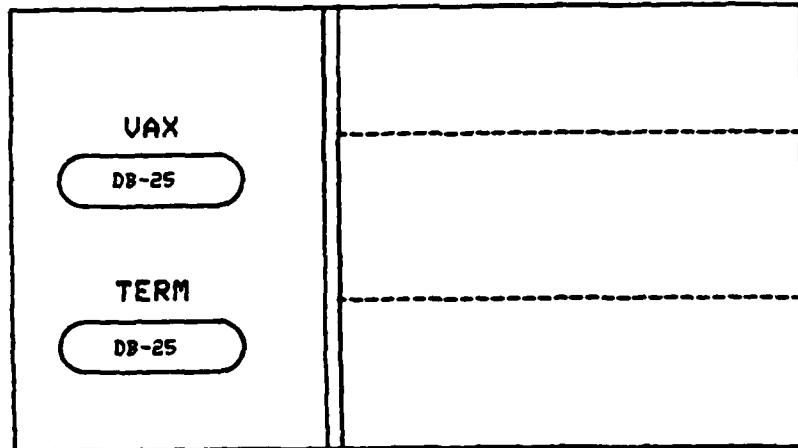


Figure 2. EVM connections

These settings need not be re-entered unless the EVM RESET is asserted twice in a row or the power is removed from the EVM.

After the above initialization of the EVM is completed, the user 'toggles' the system to communicate with the VAX VMS system. From VMS, DICES is initiated by the following sequence:

?ctrl t<cr> ...Toggles system to VAX VMS

Welcome to the AFIT/ENG Information Sciences Laboratory
(VAX/VMS version 4.2)

Last interactive login on Thursday, 10-OCT-1985 19:13
10-OCT-1985 19:15

Account: ti<cr> ...account name which contains
DICES programs

Password: ee664 ...Password

%set def [.dices] ...VMS subdirectory which
contains DICES programs
(Note: % is shown as VMS
prompt in place of dollar
sign)

%run dices ...execute DICES from VMS

DICES responds with the main menu as follows:

DIGITAL INTERACTIVE CONTROLLER EVALUATION SYSTEM

(DICES)

OPTION:

- 1. SYSTEM PERFORMANCE SPECS**
- 2. MATHEMATICAL MODEL DEVELOPMENT**
- 3. PLANT SIMULATION**
- 4. CONTROLLER DESIGN**
- 5. IMPLEMENT CONTROLLER**
- 6. CLOSED-LOOP PERFORMANCE TESTING**
- 7. REPORT GENERATION**
- 8. END PROGRAM**

The user is now free to select any of the options that do not require hardware that is not connected to the system. All options except option 6 and the part of option 5 that downloads object code to the EVM are available.

4.3 MINIMUM SYSTEM FOR IMPLEMENTATION AND TEST OF CONTROLLERS

This section discusses the addition of the TR-48 Analog Computer, Wavetek 172B Signal Generator, and B/K 2032 System Analyzer to DICES to give it full capability.

4.3.1 EVM ANALOG CONNECTIONS

The EVM unit contains (in addition to the TMS32010 microprocessor) a Digital-to-Analog (D/A) and Analog-to-Digital (A/D) converter which must be connected to the analog computer system in order to allow the filter to read the system variables. This step requires some knowledge of the operation of an analog computer in order to determine the location of the plant input/output and the input/output of the summing junction (error signal).

4.3.1.1 CASCADE CONTROLLER

For a cascade controller, the connector on the front of the EVM labelled 'INPUT' is connected to the appropriate location on the analog computer. In most cases this will be the output of the summing junction. The connector labelled 'OUTPUT' is connected to the appropriate location on the analog computer. This normally is the point where the output of the summing junction would have gone to if no controller were in the system (see figure 3).

4.3.1.2 FEEDBACK CONTROLLER

For a feedback controller, the connector labelled 'INPUT' is connected to the appropriate location on the analog computer. In most cases this will be the output of the plant. The connector labelled 'OUTPUT' is connected to the appropriate location on the analog computer. This

is normally the input to the summing junction where the output of the plant would have been fed back to if no controller were in the system (see figure 4).

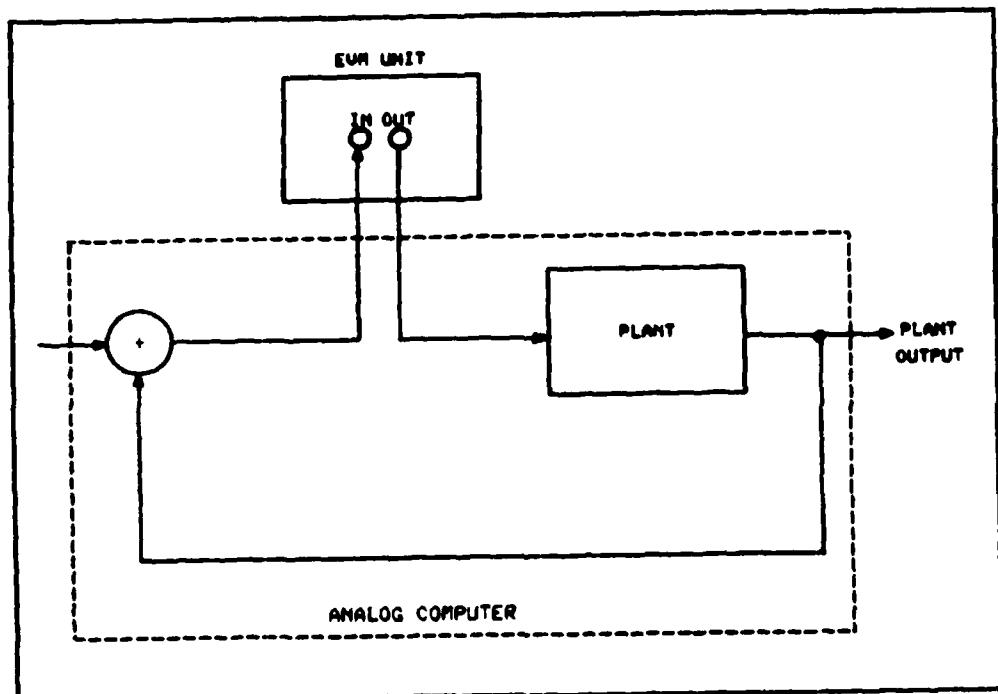


Figure 3. Cascade Controller Connections

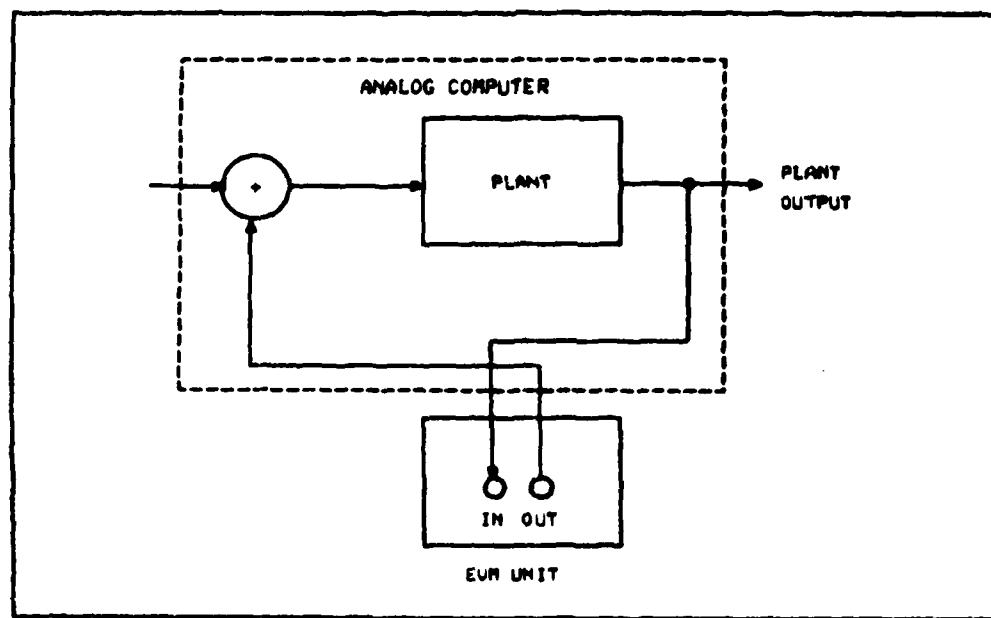


Figure 4. Feedback Controller Connections

4.3.2 OTHER EQUIPMENT

This section discusses the connection of the B/K 2032 System Analyzer, TR-48 Analog Computer System, and Wavetek 172B Signal Generator.

The B/K 2032 System Analyzer is used to make frequency and time response measurements on the simulated closed-loop control system. It is assumed that the user has wired the analog computer to simulate a plant and has an input and output point to make the following connections to (See attachment 1 for example problem).

The B/K 2032 has four interfaces that must be connected by the user.

1. Channel A Input
2. Channel B Input
3. External Trigger Input
4. Signal Generator Output

When each of these connections are made, all the connections for the other instruments will also be made in the process. The connectors on the B/K are located below the pull-out drawer. The Wavetek OUTPUT connector is located at the rear of the instrument (see figure 5).

Each of the four main B/K connections will be addressed individually in the following section.

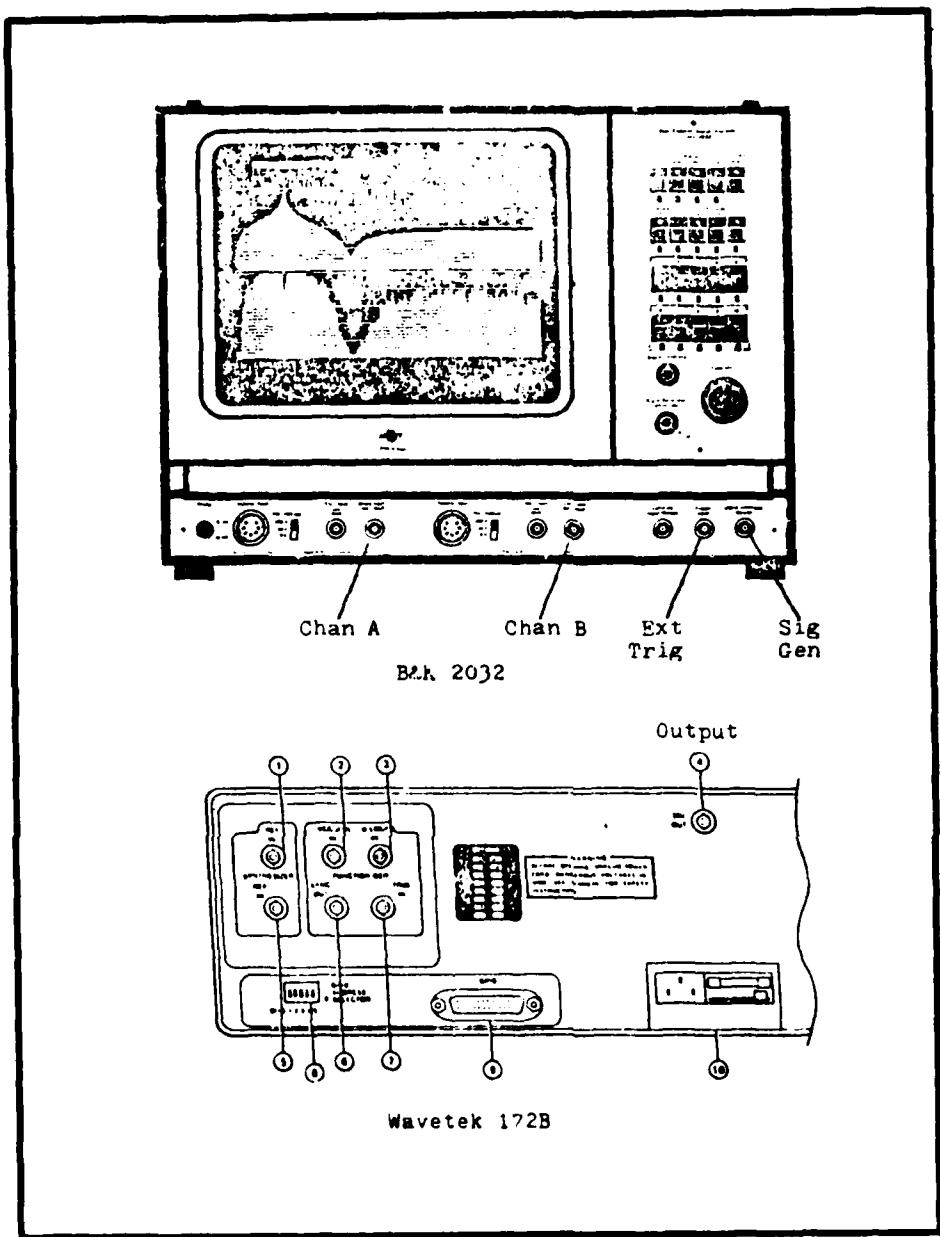


Figure 5. Location of Connections

4.3.2.1 B/K 2032 DETAILED CONNECTIONS

Step 1 Channel A. The B/K 2032 Channel A connector is connected to the input of the summing junction of the analog computer. A BNC 'Tee' connector should be used on the Channel A connector, as another cable must also later be connected to this point (See figure 6).

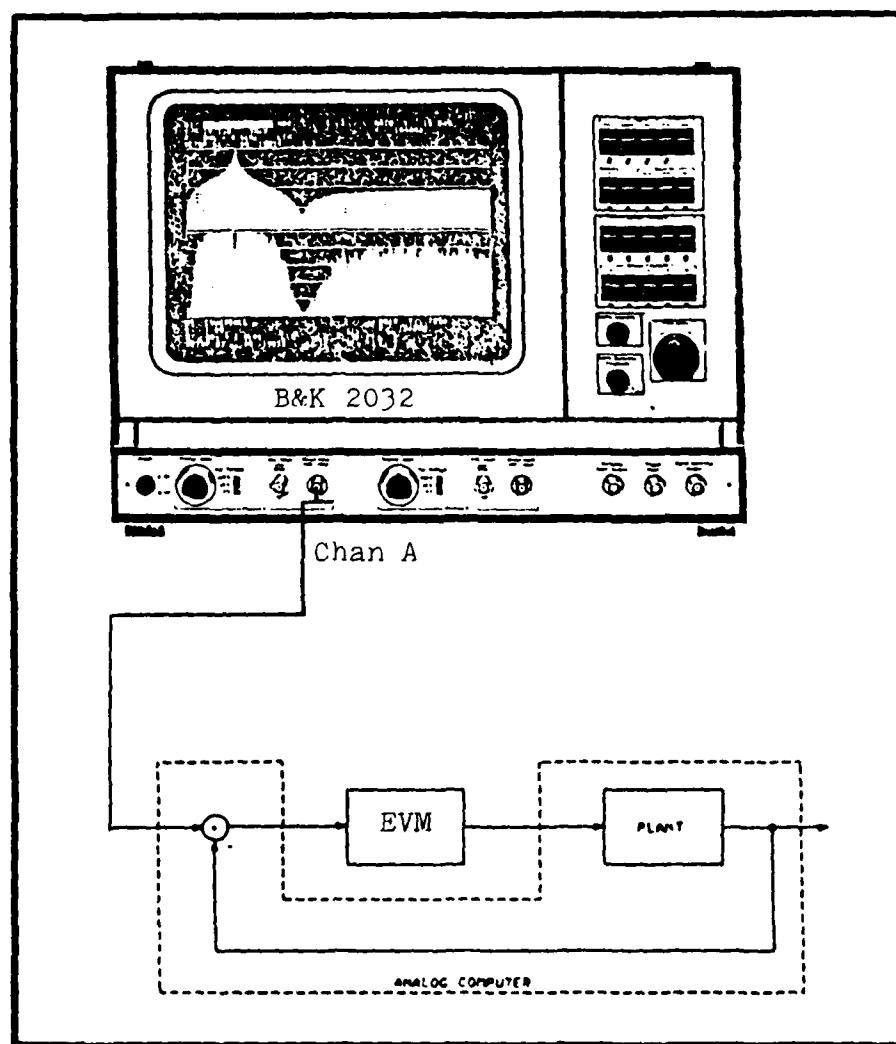


Figure 6. Channel A Connection

Step 2. Channel B. The B/K 2032 Channel B connector is connected to the analog computer plant output. This is the simulated plant output which represents the physical variable of the plant's output.

A length of BNC cable should be connected to the analog computer output with the 'low side' of the cable connected to a common terminal on the TR-48 (see figure 7).

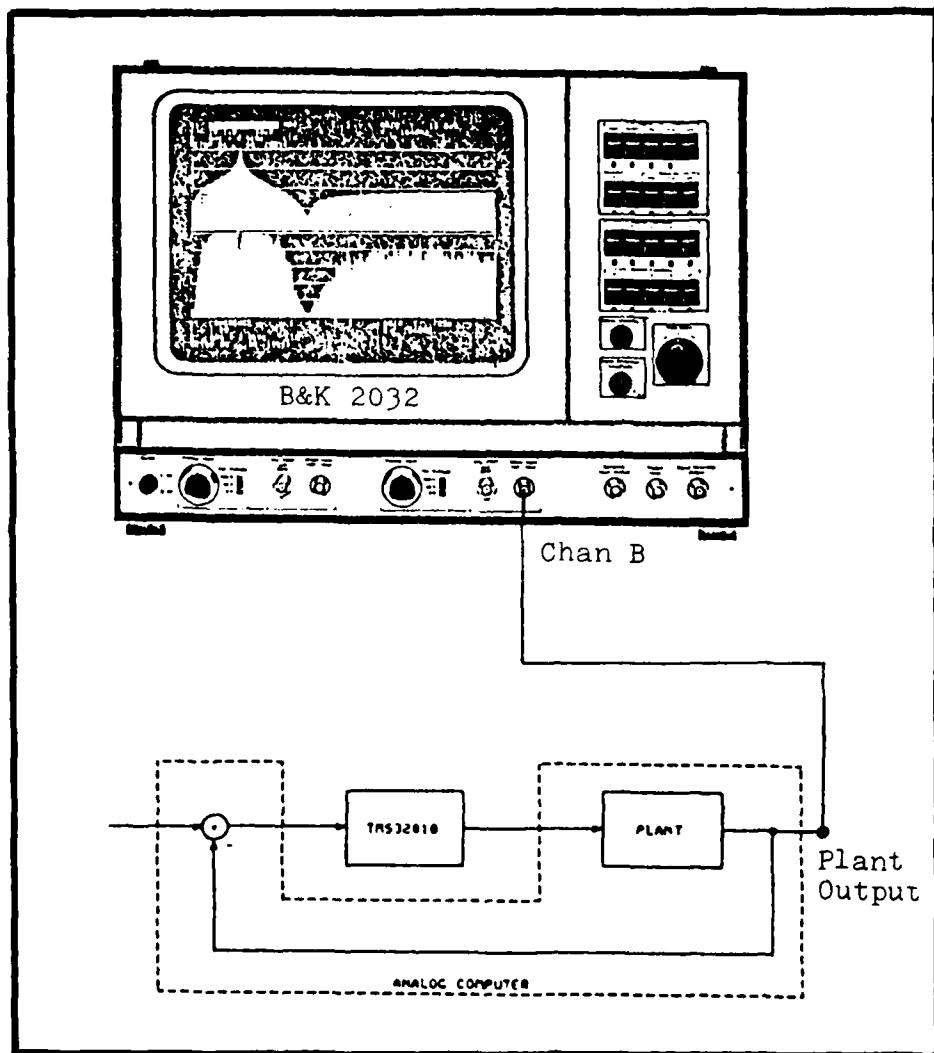


Figure 7. Channel B Connection

Step 3 External Trigger. The output of the Wavetek 172B is connected to the External Trigger Input of the B/K 2032 and to the summing junction of the analog computer in this section (see figure 8). The Wavetek 172B output connector is located at the rear of the instrument and is labelled '50 Ohms Out'. This cable should be connected to the B/K 2032 External Trigger Input connector using a BNC Tee connector. From the Tee connector, a cable is connected to the summing input of the analog computer.

IMPORTANT NOTE

A 50 ohm BNC termination must be connected to the output of the Wavetek 172B to assure accurate voltage output.

Step 4 Signal Generator Output. From the Tee connector on the Channel A input on the B/K 2032, connect a cable to the B/K signal generator output connector. Note that there are now two external inputs to the summing junction of the analog computer (see figure 9).

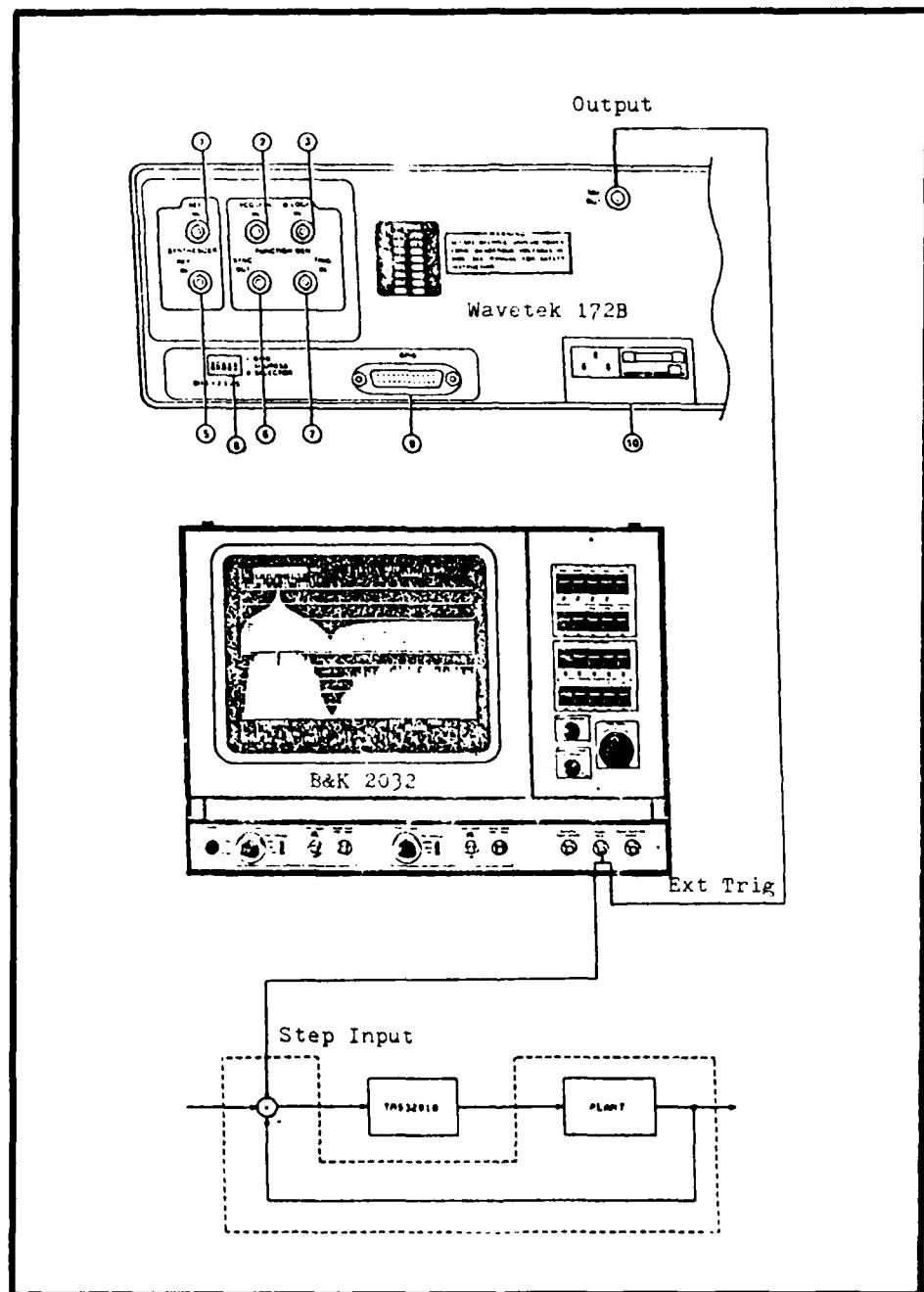


Figure 8. External Trigger Input

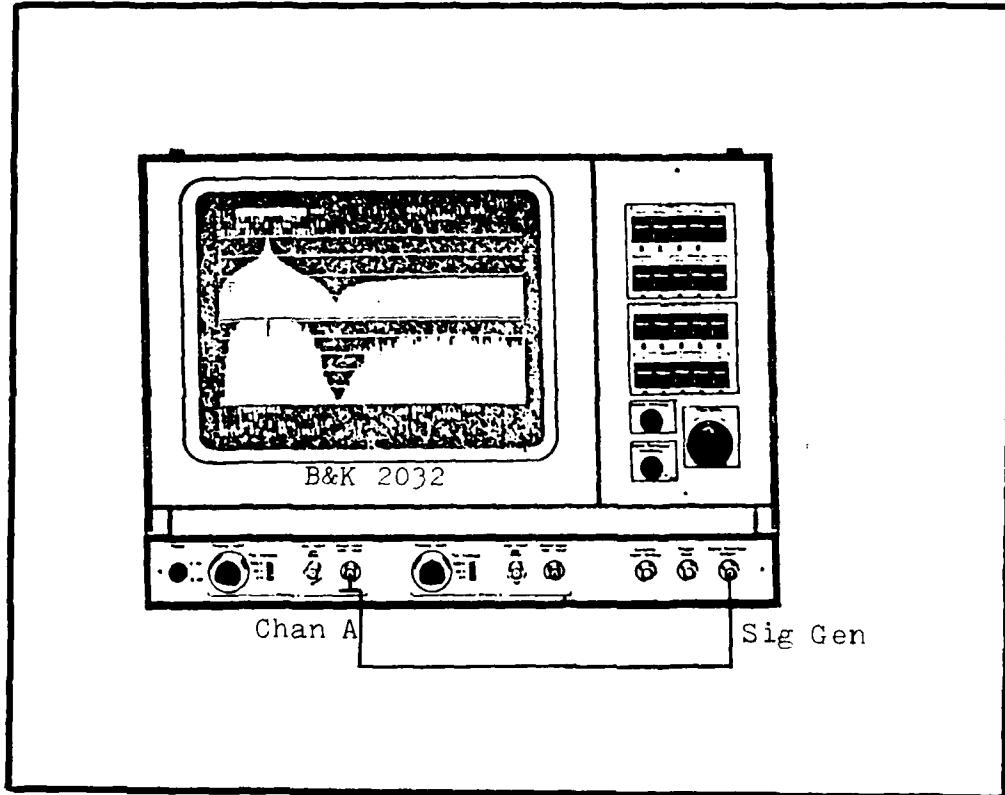


Figure 9. Signal Generator Connection

The final interconnection diagram is shown below:

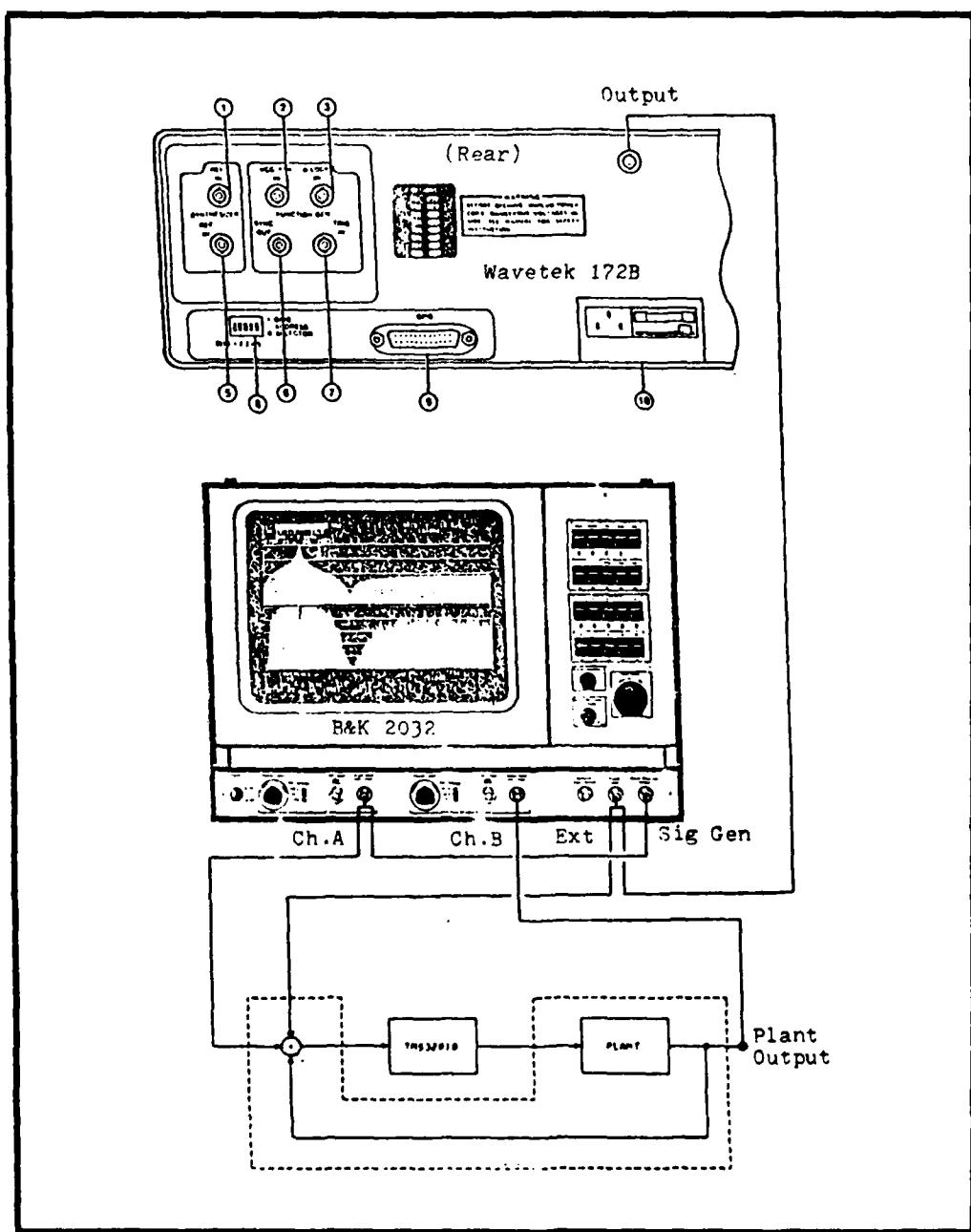


Figure 10. Interconnection Diagram

4.3.3 IEEE-488 INTERFACE BUS

The IEEE-488 bus is available from the National Instruments GPIB11-2 Interface Card in the ISL VAX 11/780. This bus is extended from the VAX 11/780 to 30 meters by using two National Instruments GPIB-100 interface extenders.

The cable from the VAX interface card is connected to GPIB-100 Unit 1 at the IEEE-488 connector. A rear view of the GPIB-100 is shown in figure 11. The RS-422 extension cable is then connected to the rear of Unit 1. The other end of the RS-422 cable is connected to the appropriate connector on GPIB-100 Unit 2 as shown in figure 12. The push-button switch labelled 'TALKER ONLY' on the GPIB-100 units should be in the out (not talker only) position.

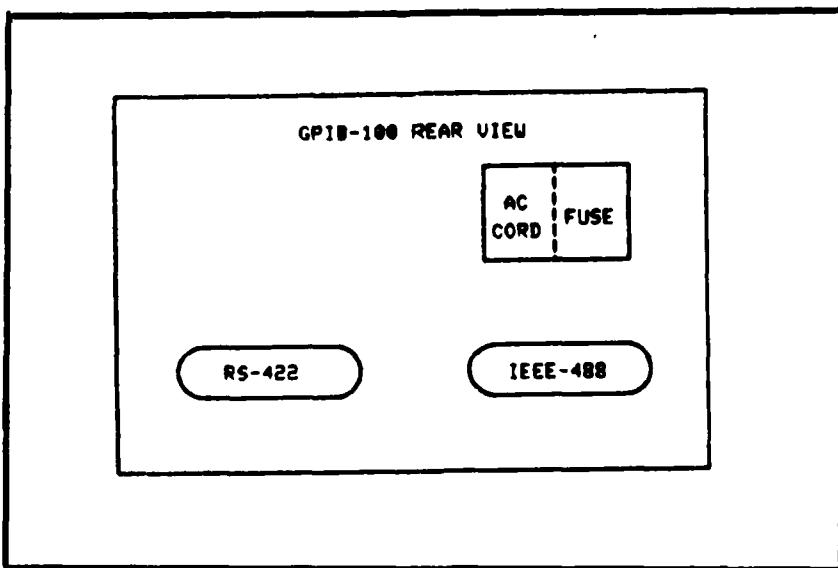


Figure 11. GPIB-100 Bus Extender Rear View

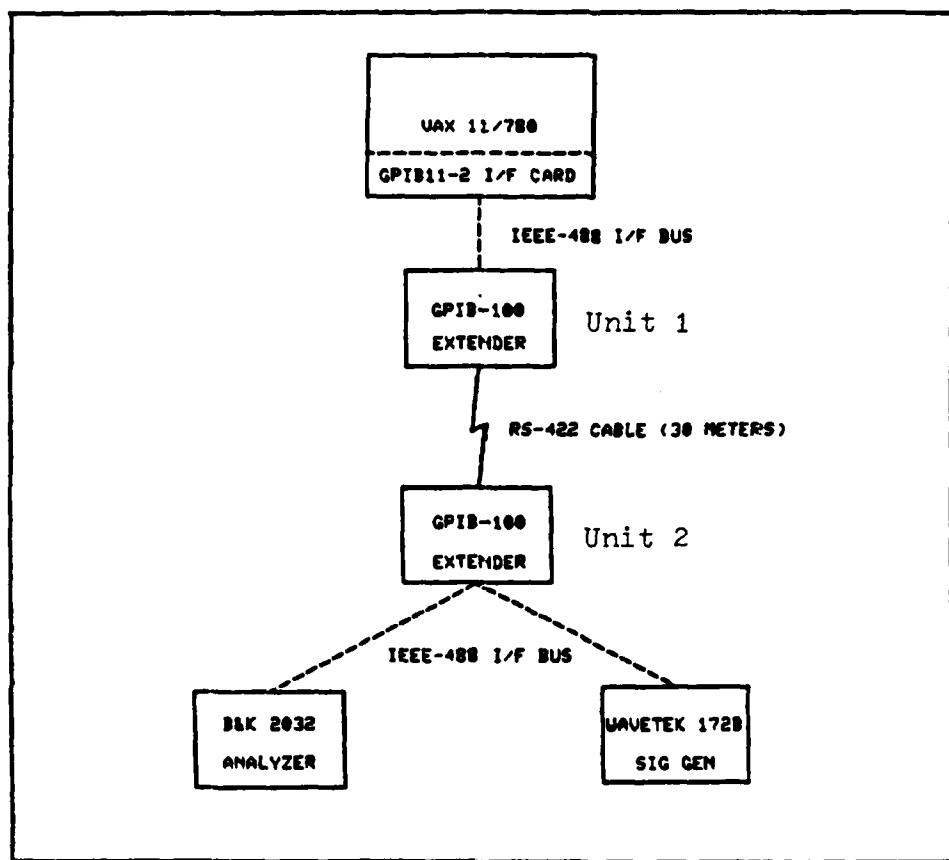


Figure 12. IEEE-488 Bus Connections

4.3.3.1 B/K 2032 and Wavetek 172B IEEE-488 Connection. The Wavetek 172B Generator and B/K 2032 System Analyzer are connected to the GPIB-100 Unit 2 IEEE-488 connector to provide remote programming capability from the VAX 11/780. An IEEE-488 cable is simply connected from each instrument's IEEE-488 connector (see figure 13) to the Unit 2 IEEE-488 bus connector such that one cable connector 'piggybacks' on the other.

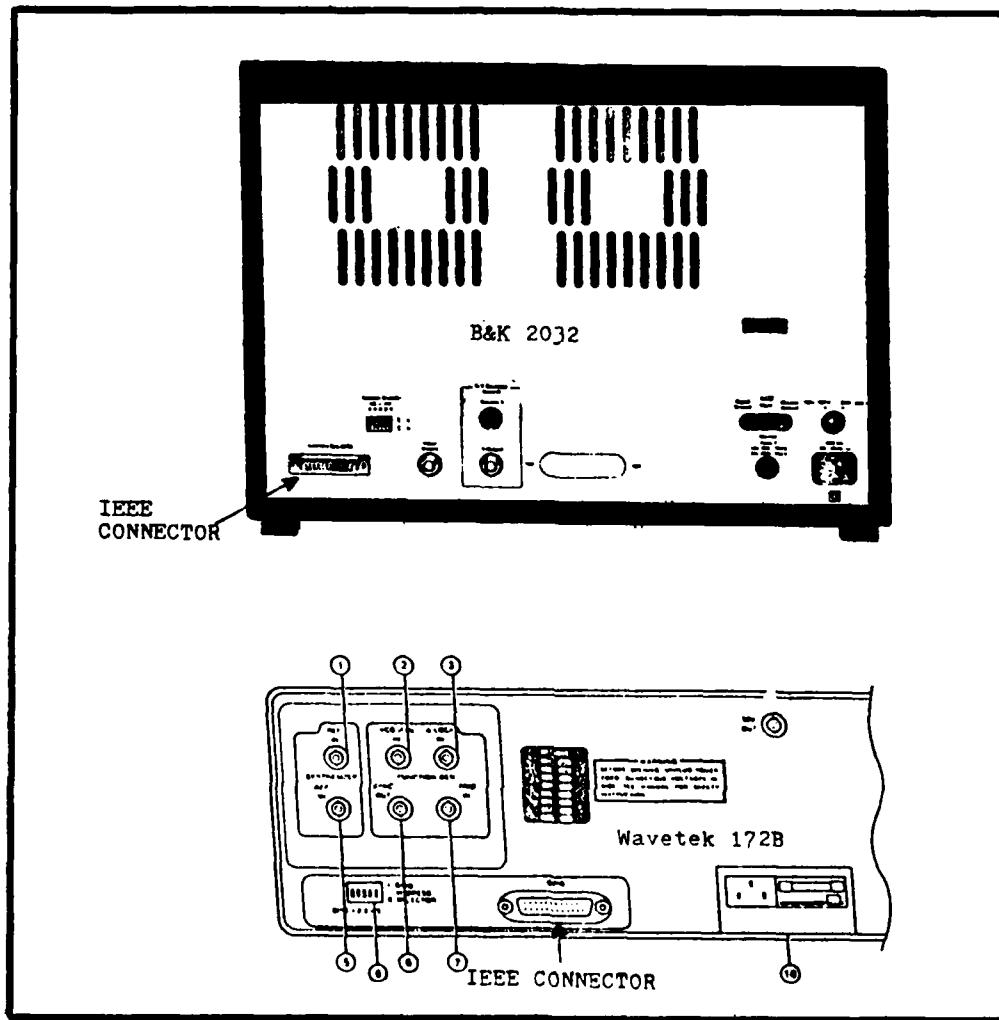


Figure 13. Instrument IEEE-488 Connections

4.3.4 POWER CONNECTIONS.

Appropriate power cables should be connected to each instrument and power applied to each instrument. Allow approximately ten minutes warm-up time for the analog computer and several minutes for the other test equipment.

5.0 FUNCTION DESCRIPTIONS

This section describes DICES' capabilities and limitations.

5.1 SYSTEM PERFORMANCE SPECIFICATIONS

Option 1 on the main menu is simply a message stating the importance and necessity to obtain performance specifications for the system under study.

DICES does not assist in this function in any way.

5.2 MATHEMATICAL MODEL DEVELOPMENT

Option 2 on the main menu is a message informing the user of the need to develop a 'good' mathematical model of the system under consideration.

DICES does not assist in this function in any manner.

5.3 PLANT SIMULATION

Option 3 on the main menu provides references to assist in the 'patching up' of the mathematical model on the TR-48 analog computer. This is a critical step in the testing of the closed-loop system. A problem in this step will not permit inserting and testing the digital controller into the closed-loop system.

5.4 CONTROLLER DESIGN

Option 4 permits the use of the control systems CAD package ICECAP. This program is used to design and analyze the digital controller required to make the closed-loop system meet specifications.

When option 4 is selected, DICES automatically calls ICECAP and displays the standard ICECAP prompts and displays. Discussion of ICECAP is beyond the scope of this manual and the ICECAP users manual should be consulted for further information.

5.4.1 SAVING CONTROLLER DESIGN

When the design process is complete, the final digital compensator design MUST be transferred to the ICECAP variable HTF. This variable is used to pass the design information back to DICES for implementation. The storage of the compensator design in HTF is accomplished using

>COPY (variable containing design) HTF

COPY COMPLETE

...Response of ICECAP

where the variable containing the compensator design may be GTF, OLTF, CLTF, etc.

5.4.2 LEAVING ICECAP

To return to DICES simply type the ICECAP command STOP. This stores the design parameters into a data file

and gives a sign-off message. DICES is automatically re-entered upon completion of this command and the main menu displayed.

5.5 IMPLEMENT CONTROLLER

Option 5 reads the design parameters of the digital compensator and implements the compensator as a cascade of second-order 3D filter sections. The 3D structure is a non-canonical direct implementation of a linear, constant coefficient difference equation. These sections are formed from a 'near optimal' pairing algorithm that picks pairs of poles and zeros to form each second-order section in an effort to minimize the effects of coefficient quantization.

The structure of each section of the compensator is as follows:

$$G_c(Z) = \frac{b_0 + b_1 Z^{-1} + b_2 Z^{-2}}{1 - a_1 Z^{-1} - a_2 Z^{-2}} \quad (1)$$

where the coefficients are formed by truncating or rounding the 16-bit integers that represent them.

5.5.1 READ DESIGN PARAMETERS

Option 1 of the IMPLEMENT CONTROLLER menu causes the design parameters of the digital controller to read into

DICES. Note that an eighth-order over eighth-order filter is the current limitation of DICES. This must be accomplished before any of the remaining options from this menu are selected. An error message will be displayed if this is not done.

5.5.2 SELECT ROUNDING OR TRUNCATION OF COEFFICIENTS

Option 2 of this sub-menu allows selection of whether round-off or truncation will be used to quantize the filter coefficients into 16-bit integer representations within the TMS32010.

5.5.3 PAIR FILTER POLES AND ZEROS

Option 3 of this sub-menu causes the filter parameters that were read into DICES to be paired into second-order sections. A 'near optimal' pairing algorithm is used to minimize effects of coefficient quantization.

The current version of DICES restricts the controller gain such that for each second-order section, any of the coefficients cannot exceed a value of .999969. This restricts the location of the zeros and poles so that when multiplied together, a coefficient greater than that stated above does not occur. This limitation was required because of time constraints while developing the system.

5.5.4 GENERATE TMS32010 OBJECT CODE

Option 4 invokes the Texas Instruments TMS Assembler

to assemble the filter source code generated by DICES. The source file generated by DICES is called 3DFILT.THS. This name must be entered when asked for by the assembler. Each file requested after the initial input can be entered by a carriage return, as the assembler uses default names tied to the initial file name 3DFILT.THS. When assembly is complete, the object file is contained in file 3DFILT.MPO. This file name must be used later for downloading the object code into the EVM.

5.5.5 DOWNLOAD OBJECT CODE

Option 5 allows the downloading of the digital filter object code to the EVM for execution on the TMS32010. DICES returns to the VMS system to accomplish this function because the VMS system must transmit the object file to the EVM.

The user types a 'control T' (shown as ~T from now on) in order to toggle out of the transparency mode of the EVM. The EVM responds with its prompt of '?' and waits for further input.

S ~T (VMS PROMPT)

? 1pm 2<cr> (TMS32010 EVM REPORT)

which puts the EVM into a Load Program Memory mode. Note

that no EVM prompt (?) is given after this command is given.

Another ~T is typed which puts the user back in touch with the VMS operating system. A VMS command of

```
% type 3dfilt.mpo ~T
```

is then given. Note that the command is followed by a ~T, NOT a carriage return. A carriage return will not work! The ~T puts the EVM back into local mode with the EVM again and sends a carriage return to the VMS system automatically. It is this carriage return which causes the VMS system to type out the file 3DFILT.MPO to the EVM.

Following the loading of the object file, a return to the VAX VMS system is required. This is accomplished by the following:

```
?~T <cr>      <toggles back to VMS>
```

```
%run dices      <executes DICES again>
```

The full main DICES menu is again presented.

5.6 CONTROLLER PERFORMANCE EVALUATION

Option 6 from the MAIN menu provides a means to test

the digital filter program in the closed-loop system. There are currently two tests implemented - step response and frequency response (magnitude and phase). The impulse response test is not implemented in the current version of DICES.

5.6.1 STEP RESPONSE

Option 1 from the PERFORMANCE EVALUATION menuselects the STEP RESPONSE TEST. DICES requests the step amplitude desired (limited to 0 to 7.5 volts) and the approximate settling time of the system. The settling time is easily obtained from the ICECAP simulations which were performed during the design process.

-T is used to communicate with the EVM in order to start execution of the filter program. When the filter program is running, the CONTROL*START button on the B/K 2032 System Analyzer is pressed to signal DICES to begin the test (see figure 14).

When the test is complete, the words 'TEST COMPLETE' will be displayed at the bottom of the B/K 2032 screen. The step response of the system will also be displayed in a full graph mode on the B/K.

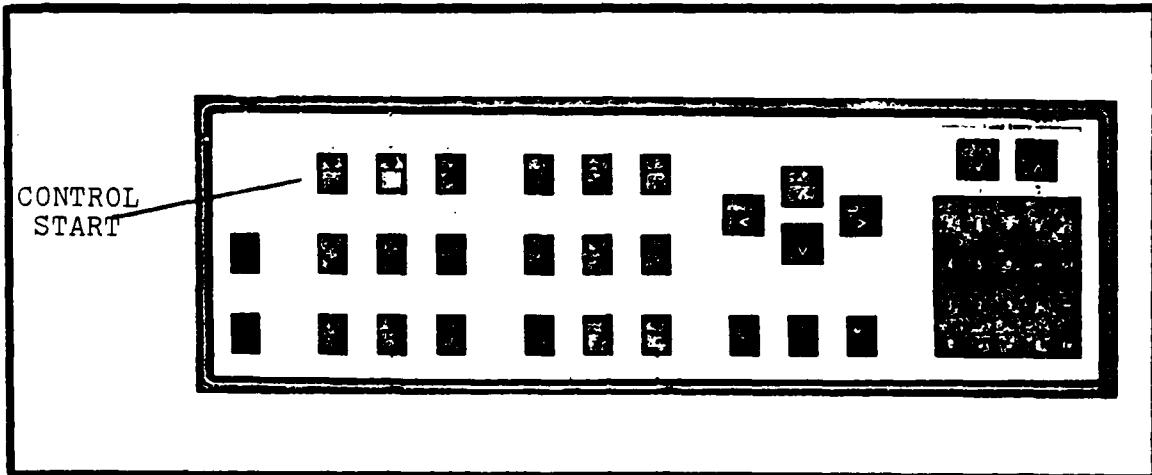


Figure 14. Location of CONTROL*START Button
on B/K 2032 System Analyzer

The cursor knob located on the panel of the B/K can now be used to move the cursor along the step response of the system to obtain information about the response (peak time, settling time, peak value, etc.).

To return to DICES, the RESET switch on the EVM must be asserted to stop the TMS32010 execution of the filter code. The EVM will return to the monitor program and display its menu. ~T and a carriage return is typed to return to the DICES main menu.

The B/K 2032 can be manipulated until the next test is performed. There is no hardcopy available for the B/K 2032 at the present time. The B/K 2032 System Manuals should be consulted for further information on other functions that may be performed after the test.

5.6.2 FREQUENCY RESPONSE

Option 2 permits the performance of a frequency response test on the closed-loop system. A display of both magnitude and phase is provided on the B/K 2032.

DICES prompts for the APPROXIMATE closed-loop bandwidth of the system (in radians/second) to allow setting the B/K 2032 properly. Note that this test may take up to SEVEN minutes for the lowest bandwidth system of about 5 - 10 radians per second.

To initiate this test, the same procedure is followed as for the step test above, which will not be repeated.

5.6.3 IMPULSE RESPONSE

The impulse test is currently not implemented on DICES.

5.7 REPORT GENERATION

This option permits selection of reports to be printed on the line printer after a filter is implemented and tested.

5.7.1 FILTER PAIRS AND QUANTIZED COEFFICIENTS

This report lists the digital filter pole and zero pairs which were matched to form each second-order 3D filter section. The values of each coefficient is given before and after quantization and the integer representation of each coefficient is given.

6.0 EXAMPLE CONTROL PROBLEM

This section presents the step-by-step procedures used to solve a control problem using DICES. This example serves as a good introduction to the use of DICES.

Note that in the following procedures, computer system responses are typed in UPPER CASE and user input is typed in lower case. The only exception to this is when a control character is typed by the user. For example , 'control T' is shown as ~T .

System Connection. The EVM must be connected as detailed in section 4.3.1. The EVM must be connected to a VT-100 terminal from the rear connector labelled TERM. The EVM must also be connected to the VAX system at the connector labelled VAX. VAX VMS Terminal TTB3 should be used as it is configured for 4800 baud and several features such as NOWRAP are permanently disabled. Consult the ISL chief systemsengineer or VAX system manager for assistance if

required. This is the minimum system configuration to sign on to the VAX system.

The power switch on the front panel of the EVM unit should be switch to ON. The following sequence must now be executed to properly configure the system.

1. Toggle the front-panel RESET switch to reset EVM system.

2. Hit RETURN on the VT-100 terminal

3. After the EVM start-up message display, type the following:

? xon<cr>

? init ext<cr>

? comm

COMM <~C> CTRL T<cr> Note: EVM types COMM <~C>

User types ~T<cr>

These commands to the EVM monitor program initialize the system hardware and software to communicate with the Analog Interface Board and the VAX VMS system. A brief description of the effect of each command is given below.

The 'xon' command turns on the XON/XOFF protocol which controls the sending and receiving of data between the VAX and EVM.

The 'init ext' command sets the clock used on the AIB to an external clock which is supplied from the processor board in the EVM.

The 'comm' command sets the EVM to respond to a ~T to go into the TRANSPARENCY MODE of the EVM. This replaces the default value of ~C which is normally used by EVM users. This is because the VMS system uses ~C to break out of a program. This would have the effect of constantly toggling the user in and out of the TRANSPARENCY MODE of the EVM.

VAX Sign-On Procedure

Once the EVM intitialization procedure is complete, the user is ready to sign on to the VAX system. The first command given is to toggle the EVM into the TRANSPARENCY MODE. This is accomplished by typing

? ~T

and hitting a <cr>.

The VAX system responds with the system prompt as follows:

Welcome to the AFIT/ENG Information Sciences Laboratory
(VAX/VMS version 4.2)

25-AUG-1985 09:05

The user responds with the account name and password:

Username: ti

Password: ee664

The user then types the command to run the DICES program:

```
% set def [ti.dices]          ...change to DICES directory  
% run dices                  ...run DICES program
```

DICES presents the opening menu:

DIGITAL INTERACTIVE CONTROLLER EVALUATION SYSTEM (DICES)

OPTIONS:

1. SYSTEM PERFORMANCE SPECS
2. MATHEMATICAL MODEL DEVELOPMENT
3. PLANT SIMULATION
4. CONTROLLER DESIGN
5. IMPLEMENT CONTROLLER
6. CLOSED-LOOP PERFORMANCE TESTING
7. REPORT GENERATION
8. END PROGRAM

6.1 DETERMINE SYSTEM PERFORMANCE SPECIFICATIONS

Selecting option 1, the user is directed to determine the system performance specifications of the system under study. This step is performed by the user prior to using DICES. The user/designer analyzes the requirements of the system and quantifies them using standard control system figures of merit.

For this example, the system performance requirements are as follows (for a unit step input):

- < 20% peak overshoot (M_p)
- < 7.5 seconds peak time (t_p)
- < 15 seconds settling time (t_s)

The main menu is again presented to the user.

6.2 DEVELOP PLANT MODEL

Option 2 is selected which directs the user to develop the mathematical model of the plant to be controlled. This step is performed external to DICES. The mathematical model for the plant (in this case the amplifier and motor) is developed and used as the basis for the later design of the controller.

The motor model is shown in figure 15. It is now necessary to determine the transfer function of the motor model.

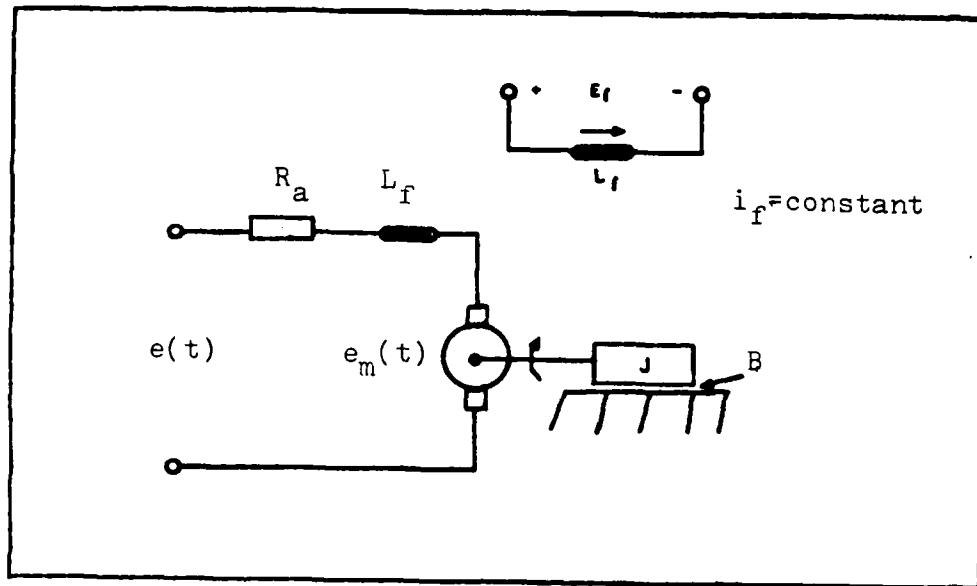


Figure 15. Servomotor system

To determine the transfer function of the motor, the differential equations that describe the system dynamics are written. Note that initial conditions are neglected since a transfer function is being obtained.

The back emf of the motor, e_m , is given by

$$e_m(t) = K_b \omega = K_b \dot{\delta} \quad (2)$$

where

δ is the motor shaft position

ω is the shaft angular velocity

K_b is a motor-dependent constant

and

$$\tau(t) = J \ddot{\delta}(t) + B \dot{\delta}(t) \quad (3)$$

where

$\tau(t)$ is the torque developed by the motor

J is the moment of inertia connected to shaft

B is the total viscous damping

$\dot{\delta}(t)$ is shaft angular position time derivative

$\ddot{\delta}(t)$ is shaft angular velocity time derivative

The torque developed by the motor is given by

$$\tau = K_t i(t) \quad (4)$$

where

$i(t)$ is the armature current

K_t is a constant

The final equation required is the loop voltage equation which is

$$e(t) = i(t)R_a + e_m(t) + L_a \dot{i}(t) \quad (5)$$

where

$e(t)$ is the input voltage to the motor

R_a is the armature resistance

L_a is the armature inductance

$\dot{i}(t)$ is the time rate-of-change of $i(t)$

These four equations are now solved for $\delta(t)$, the output, as a function of $e(t)$, the input.

Taking the Laplace transform of equations (2) through (5) and neglecting initial conditions yields:

$$E_m(s) = K_b s \delta(s) \quad (6)$$

$$\tau(s) = Js^2 \delta(s) + Bs \delta(s) \quad (7)$$

$$\tau(s) = K_t I(s) \quad (8)$$

$$E(s) = I(s)R_a + E_m(s) + L_a I(s)s \quad (9)$$

From equations 6 and 9

$$E(s) = I(s)R_a + L_a s I(s) + K_b s \delta(s) \quad (10)$$

and solving for $I(s)$ produces

$$I(s) = \frac{E(s) - s \delta(s)}{L_a s + R_a} \quad (11)$$

From equations (7), (8), and (11)

$$Js^2 \delta(s) + Bs \delta(s) = \frac{K_T [E(s) - s \delta(s) K_b]}{L_a s + R_a} \quad (12)$$

$$(L_a s + R_a)(Js^2 \delta(s) + Bs \delta(s)) = K_T(E(s) - s \delta(s) K_b) \quad (13)$$

$$(L_a s + R_a)(Js^2 \delta(s) + Bs \delta(s)) + (K_T s \delta(s) K_b) = K_T E(s) \quad (14)$$

$$JL_a s^2 \delta(s) + BL_a s^2 \delta(s) + R_a J s^2 \delta(s) + \\ R_a B s \delta(s) + K_T s \delta(s) K_b = K_T E(s) \quad (15)$$

$$\frac{\delta(s)}{E(s)} = \frac{K_T}{JL_a s^2 + (BL_a + R_a J)s^2 + (R_a B + K_T K_b)s} \quad (16)$$

$$\frac{\delta(s)}{E(s)} = \frac{K_T}{s[JL_a s^2 + (BL_a + R_a J)s + (R_a B + K_T K_b)]} \quad (17)$$

$$\frac{\delta(s)}{E(s)} = \frac{K_T / JL_a}{s[s^2 + \frac{[BL_a + R_a J]s}{JL_a} + (R_a B + K_T K_b)]} \quad (18)$$

So it can be seen that the plant model is a third-order transfer function. Had the armature inductance, L_a , been neglected (as is often done), the plant model would have been second-order.

After inserting typical values for the various parameters, the plant transfer function is obtained.

$$G_p(s) = \frac{2}{s(s+1)(s+2)} \quad (19)$$

6.3 SIMULATE PLANT MODEL

The transfer function for the plant may be simulated on an analog computer as shown in figure 16.

The voltage scaling used is:

+ 3.14 volts $\Rightarrow \pi$ radians

- 3.14 volts $\Rightarrow -\pi$ radians

Step input of 1 volt = 1 radian command $e(t)_{cmd}$

After wiring the above plant model and closing the feedback loop, it can be seen that the actual simulation results closely follow that of the uncompensated theoretical results (see figures 17 and 18).

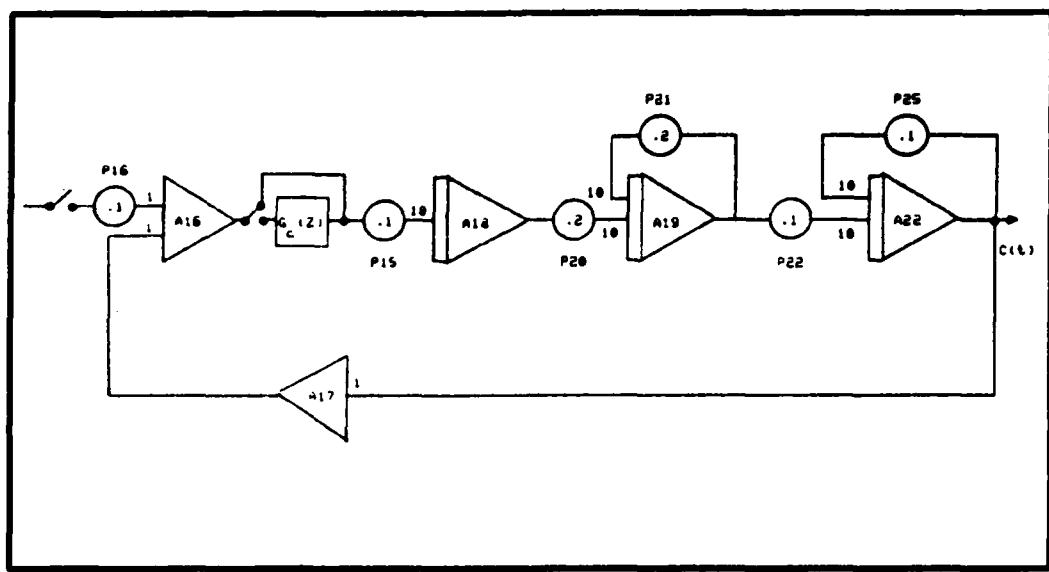


Figure 16. Analog Computer Simulation

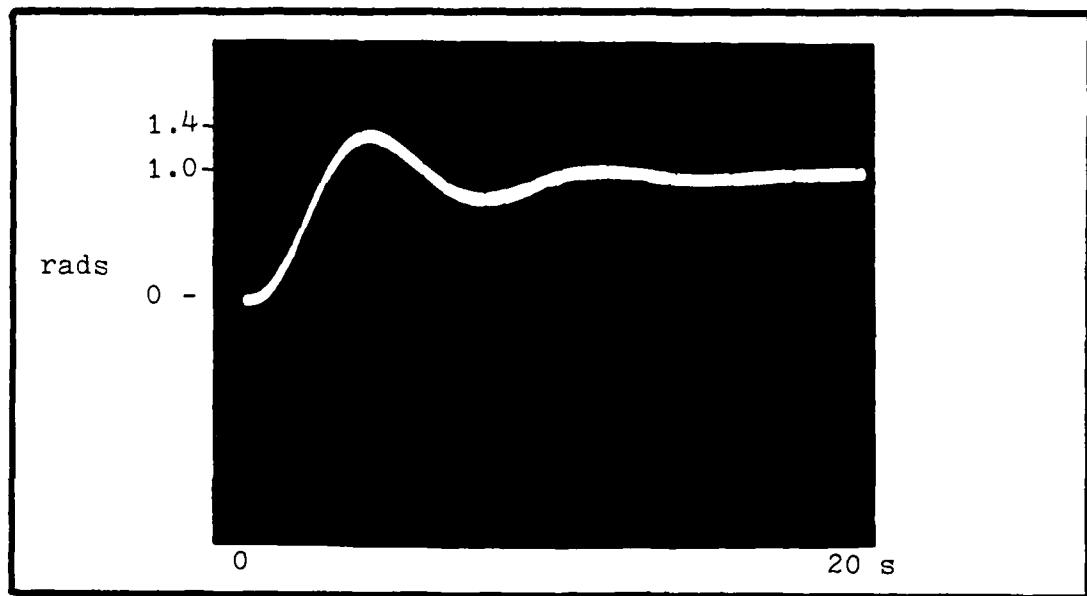


Figure 17. Uncompensated Actual Closed-Loop Simulation

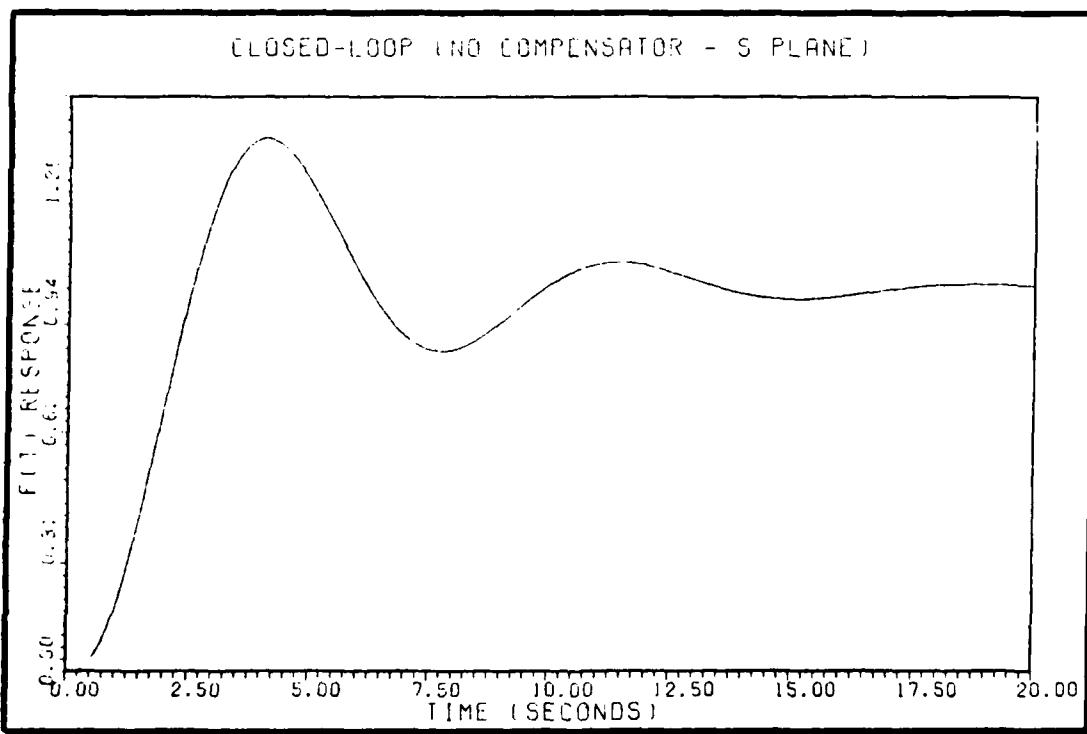


Figure 18. Uncompensated Theoretical Closed-Loop Simulation

6.3.1 ZOH AND SAMPLE PERIOD EFFECTS

After the plant was tested with no sampler or ZOH, various tests were performed that examined the effects of inserting a ZOH and sampler into the forward loop of the control system. The sample-period was varied to see the effects of lengthening this parameter.

The photographs were taken from a storage oscilloscope that was used to record the plant output voltage waveform after being subjected to a unit step input. In all cases, the theoretical response is shown along with the actual results obtained from the analog computer simulation. The block diagram of the test set-up is shown below.

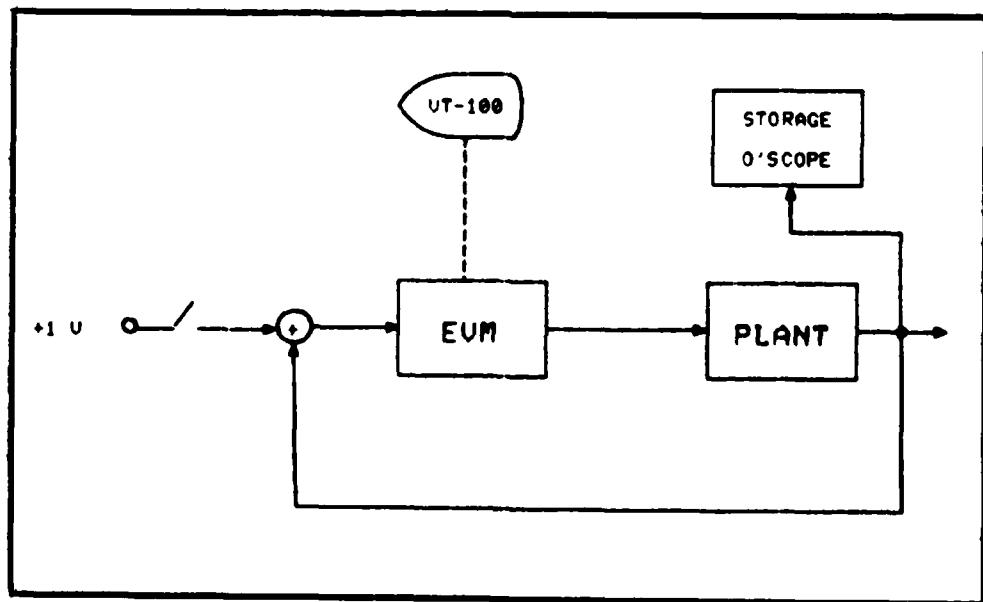


Figure 19. Sample Period Test Set-Up

6.3.2 TEST CASES

Test 1

Sample Period: .05 secs

Figures 20 and 21.

Test 2

Sample Period: .26 secs

Figures 22 and 23.

Test 3

Sample Period: .65 secs

Figures 24 and 25.

Test 4

Sample Period: 1.31 secs

Figures 26 and 27.

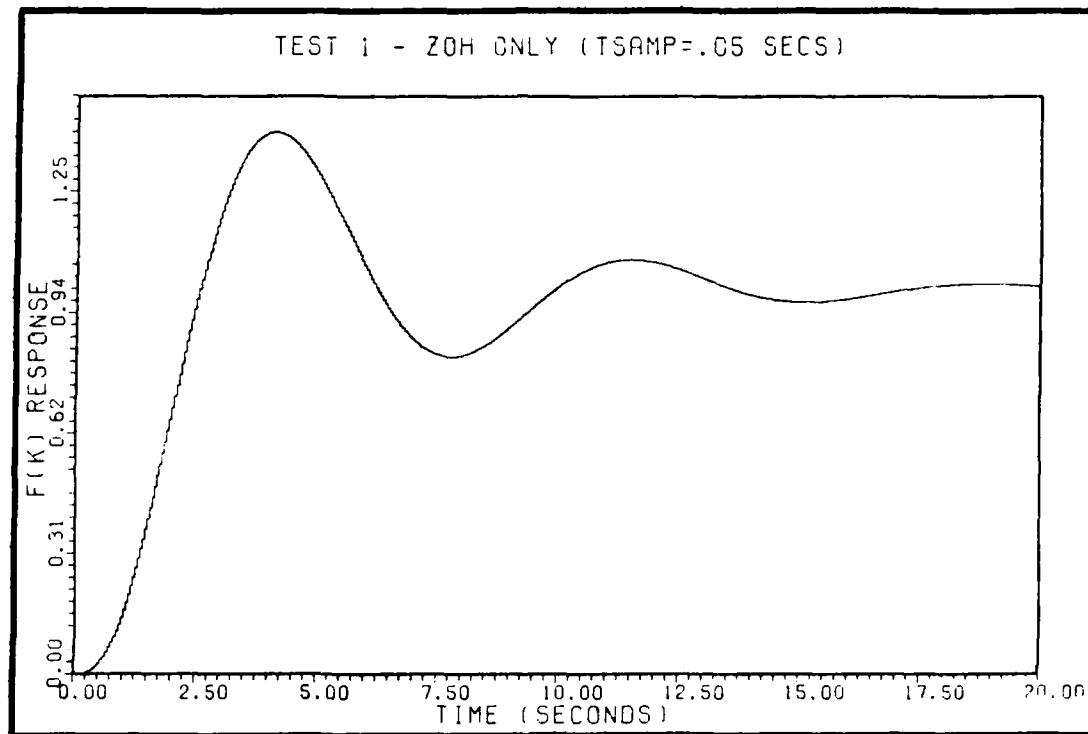
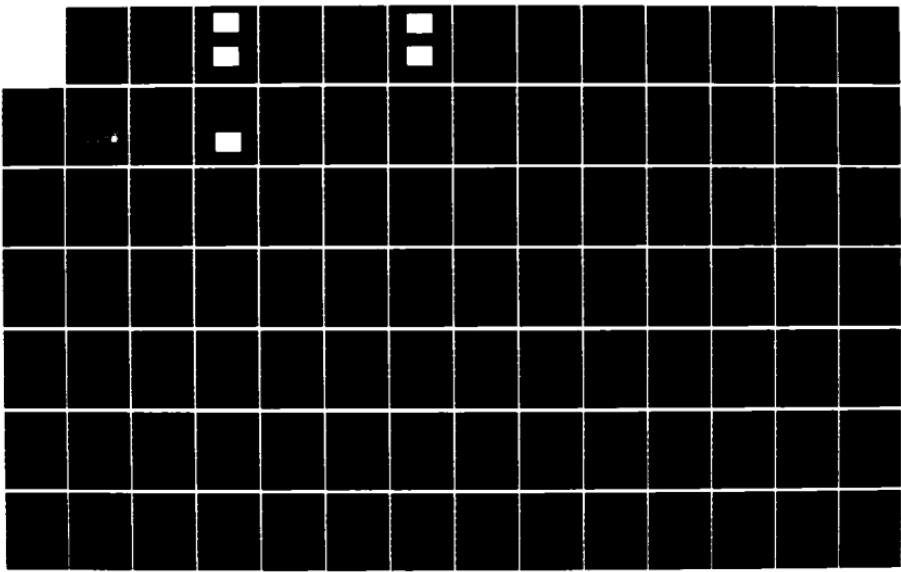
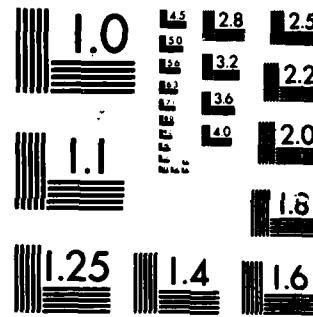


Figure 20. Analytical Response for TSAMP=.05

AD-A163 966 DEVELOPMENT OF A DIGITAL INTERACTIVE CONTROLLER
EVALUATION SYSTEM (DICES)(U) AIR FORCE INST OF TECH 4/5
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
UNCLASSIFIED S B ECKERT DEC 85 AFIT/GE/ENG/85D-13 F/G 9/5 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

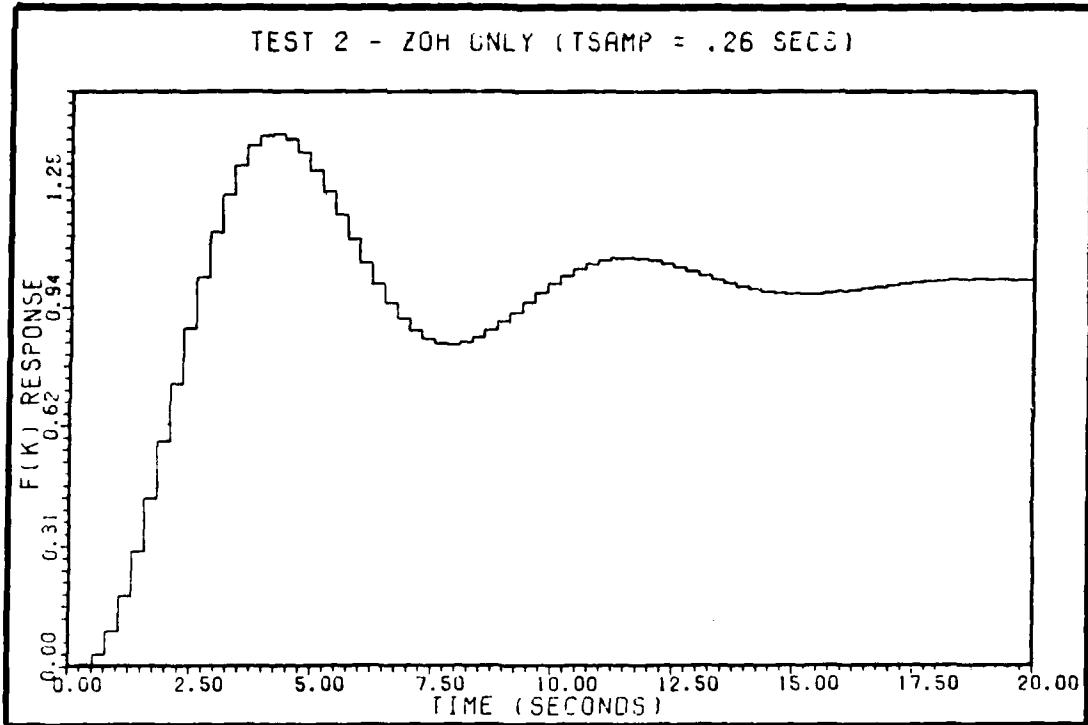


Figure 22. Analytical Response for TSAMP=.26

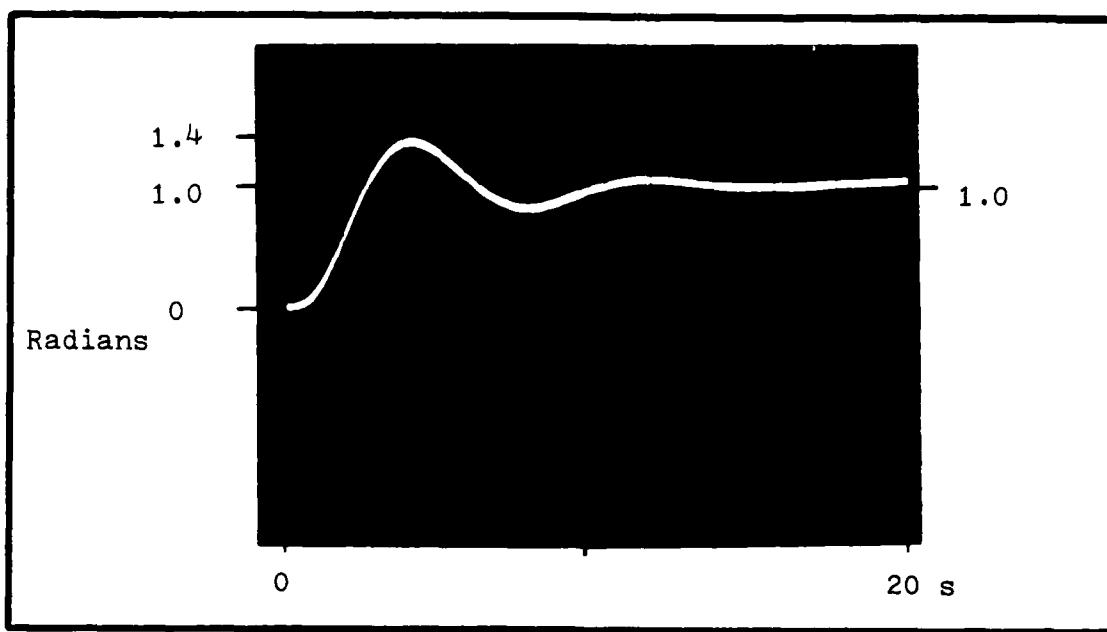


Figure 21. Actual Response for $TSAMP=.05$

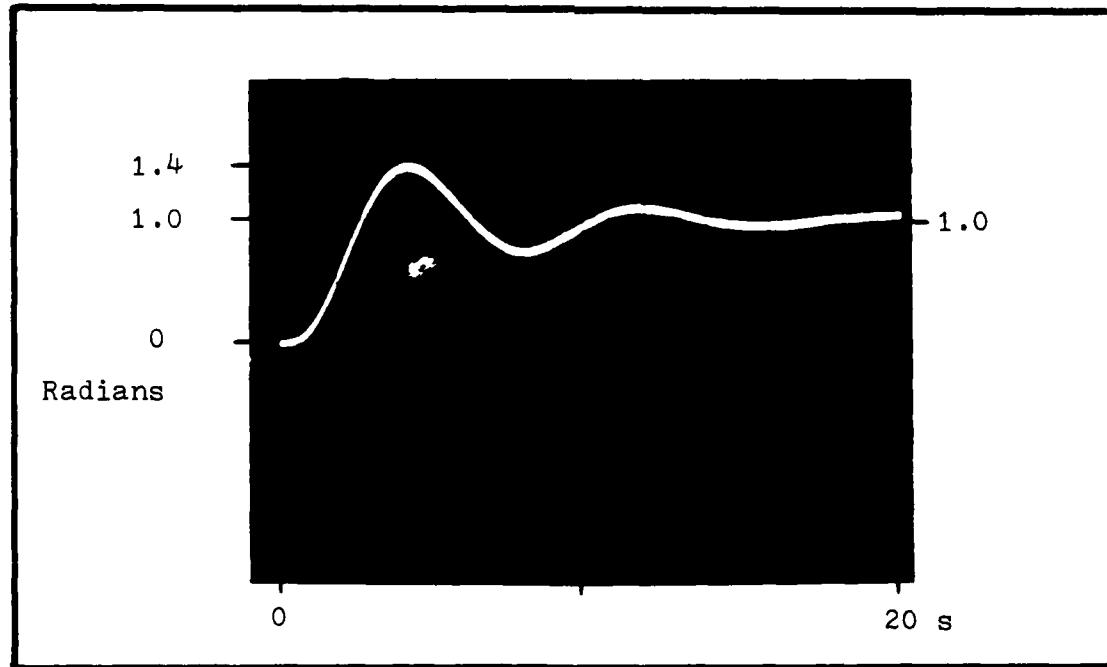


Figure 23. Actual Response for $TSAMP=.26$

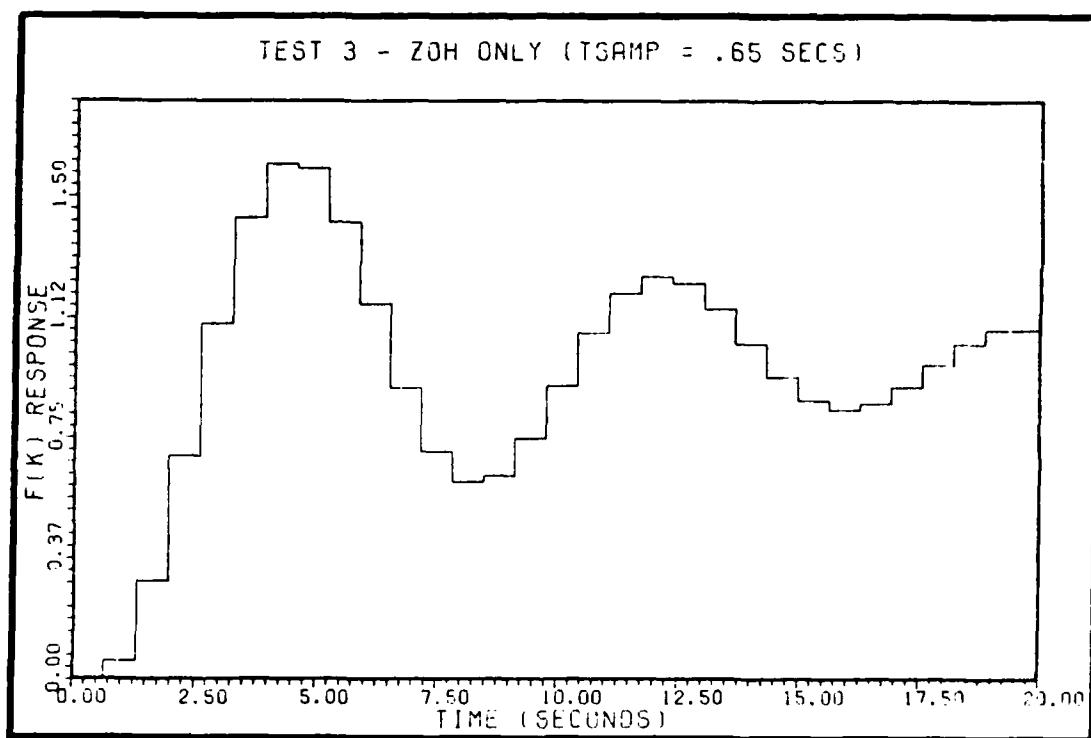


Figure 24. Analytical Response for TSAMP=.65

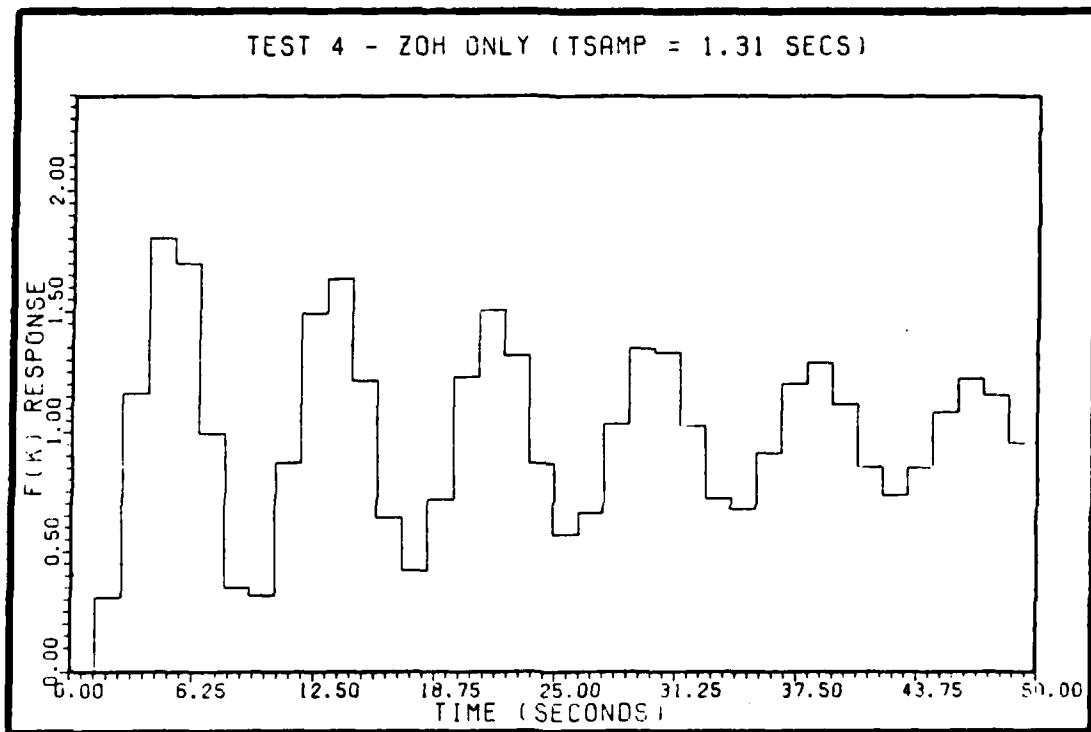


Figure 26. Analytical Response for TSAMP=1.31

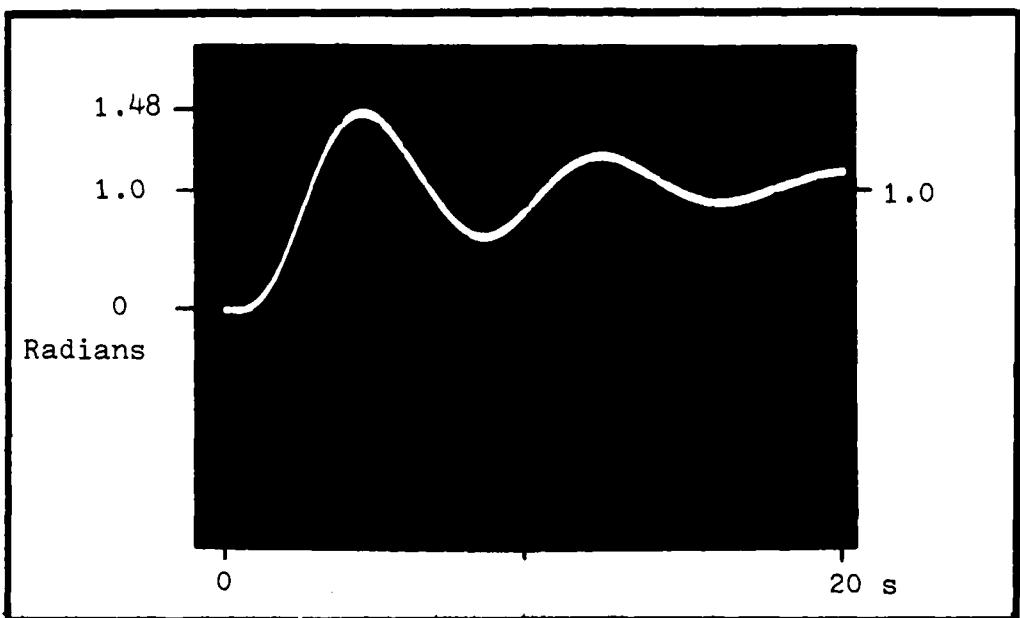


Figure 25. Actual Response for TSAMP=.65

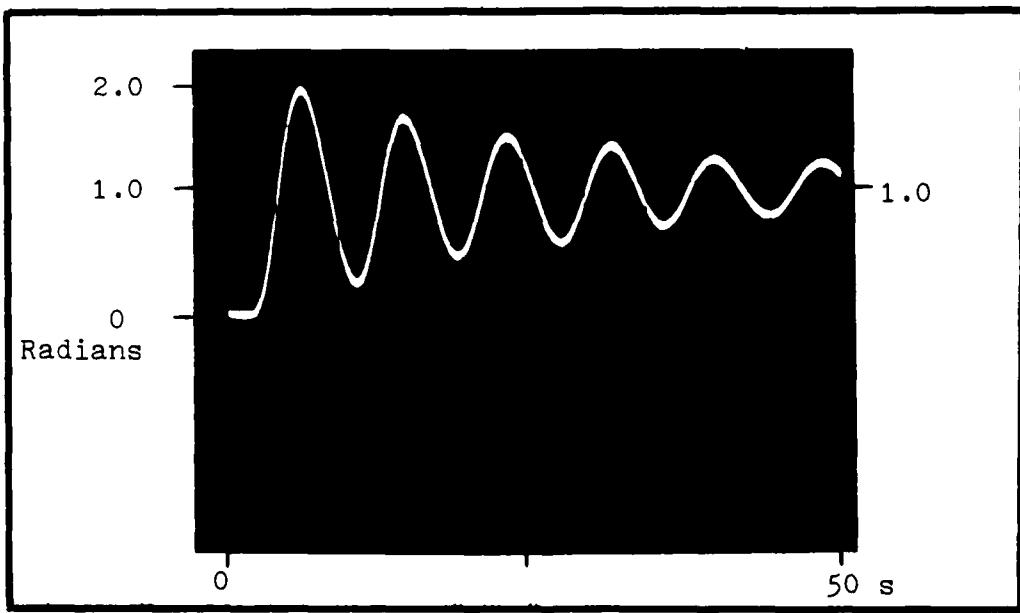


Figure 27. Actual Response for TSAMP=1.31

It can be seen that there is little effect from a sample period of .05 seconds or .26 seconds. However, when the sample-period is .65 or 1.31 seconds as shown in figures 25 and 27, the stability of the system becomes much worse.

6.4 CONTROLLER DESIGN

Option 4 from the main DICES menu is selected which transfers the user to ICECAP. From here the controller is designed to meet the system performance requirements. The details of the actual controller design are omitted for brevity. However, the plant digitization is shown. It is also noted that gain alone will not adequately correct the performance problems. A frequency response method of compensator design is used in order to produce a phase margin of 55 degrees.

The sample period is taken to be .05 seconds, which is one-tenth the fastest time-constant in the plant. The Z-transfer function of the plant transfer function is determined using the impulse invariance technique. A Zero-Order-Hold (ZOH) is correctly assumed to be present at the output of the EVM controller. This yields

$$Z[G_p(s)] = \frac{Z - 1}{Z} Z \left[\frac{2}{s^2(s + 1)(s + 2)} \right] \quad (20)$$

$$Z[G_p(z)] = \frac{.00004014(z + 3.595)(z + .25)}{(z - 1)(z - .9512)(z - 9048)} \quad (21)$$

The loop was first closed with only a sampler and ZOH to determine if there are negative effects due to the sample-and-hold process. It can be seen from the last section that the ZOH alone with a sample period of .05 seconds had no negative impact on the system response. Thus, the .05 second sample-period is a very conservative selection.

Now, using the plant Z-transfer function as the basis for the compensator design yields the following:

$$G_c(z) = \frac{.3974(z - .995)}{(z - .998)} \quad (22)$$

Forming the closed-loop system using ICECAP yields the following analytical step-response characteristics:

$$M_p = 1.25$$

$$t_p = < 8 \text{ seconds}$$

$$t_s = 12 \text{ seconds}$$

$$\delta_{ss}(t) = 1 \text{ radian}$$

which meet the requirements originally imposed on the system.

Upon completion of this step, the user enters the ICECAP command STOP which exits the ICECAP program and saves the design parameters for implementation.

6.5 FILTER IMPLEMENTATION

The user is presented with the main DICES menu following the exiting of ICECAP. The user selects option 5 which is the Filter Implementation Menu.

CONTROLLER IMPLEMENTATION

OPTION:

1. READ DESIGN PARAMETERS
2. ROUND-OFF OR TRUNCATION
3. FORM SECOND-ORDER SECTIONS
4. GENERATE OBJECT CODE
5. DOWNLOAD OBJECT CODE
6. RETURN TO MAIN MENU

Figure 28. Filter Implementation Menu

Each of the steps required to implement the controller will now be presented.

The first function executed is to read the design parameters and pair the poles and zeros into second-order sections. Selecting option 1 of the Implementation Menu produces the simple pairing of the pole and zero, since this is a first-order compensator.

DICES

```
3D DIGITAL CONTROLLER IMPLEMENTATION
=====
NUMBER OF
SECOND-ORDER SECTIONS          1
NUMERATOR ORDER                 1
DENOMINATOR ORDER               1
LOOP SENSITIVITY                .3479000
=====
UNPAIRED POLES AND ZEROS
=====
CZERO
(0.9950000,0.0000000E+00)

CPOLE
(0.9980000,0.0000000E+00)
=====
SECOND-ORDER SECTION PAIRINGS
=====
SECTION NUMBER =                1
=====
CORDPZ ZEROS(1) =   (0.9950000,0.0000000E+00)
CORDPZ ZEROS(2) =   (9999.000,9999.000)
CORDPZ POLES(1) =   (0.998000,0.0000000E+00)
CORDPZ POLES(2) =   (9999.000,9999.000)
```

Figure 29. Pole/Zero Pairing

Option 2 is then selected which allows selection of round-off or truncation of the integers which are formed during the quantization process as discussed in the design chapter. Round-off will be selected for this example problem.

Option 3 forms the second-order sections with actual coefficients and sample period integers that will be assembled into the TMS32010 object code. The coefficients and sample period integers formed as a result of this step are shown below.

```
QUANTIZATION ROUTINES
=====
NUMBER OF SECTIONS           1
HK = 0.3973999990033917
TSAMP = 5.0000001E-02
=====
SECTION NUMBER           1
=====
B0 = 0.3974363820599411
B1 = -0.3954492020447657
IB0 =      13023
IB1 =      -12957
A1 = 0.9980000257492065
IA1 =      32702
=====
TSAMP = 5.0000001E-02
ITSAMP =      419
```

Figure 30. Second-order Section Data

Option 4 assembles the source program into executable object code for the TMS32010. Upon selection of this option, a prompt will appear asking for the filename of the source code. The response must be 3DFILT.THS. After this entry, the user simply hits the RETURN key to enter the remaining default file names. Upon completion of the assembly process, the user is returned to the Implementation menu. The file 3DFILT.MPO contains the object code in TMS9900 format which permits loading by the Evaluation Module.

```
K00003DFILT90000BF900B001BB32DFBCD63B0000B7FBEBO000B7FFF800
007F1AFF      3DFILT 1
B0000B0000B0000B7FFF8000B0000B0000B0000B7FFF8000B0000B000
0B00007F21DF  3DFILT 2
B01A3B000AB0190B7FFF8000B6E00B7E01B502EB702CB7118B7E1AB688
0B67917F152F 3DFILT 3
B102EBFE00B0021B7013B7113B7F89B6880B5091BF400B0029B202BB612
CB502F7F18BF  3DFILT 4
B4829B4928B7F8BB202ABF600B0038BF900B0034B102EB4209BFE00B003
4B20097F184F  3DFILT 5
B782BB5009B7F89B6A0BB6D16B6B0AB6D15B6B09B6D14B6C13B6D18B6B1
2B6D177F100F  3DFILT 6
B7F8FB5912B6A08B6D1BB6B07B6D1AB6B06B6D19B6C11B6D1DB6B10B6D1
CB7F8F7F0BEF  3DFILT 7
B5910B6A05B6D20B6B04B6D1FB6B03B6D1EB6C0FB6D22B6B0EB6D21B7F8
FB590E7F0DBF  3DFILT 8
B6A02B6D25B6B01B6D24B6B00B6D23B6C0DB6D27B6B0CB6D26B7F8FB590
CB592D7F0E8F  3DFILT 9
B202CB782DB502DB4A2DBF900B00337F916F  3DFILT10
3DFILT 8/23/85 15:15:27   ASM320  2.1 83.076  3DFILT11
```

Figure 31. TMS32010 Assembler TMS9900 Object File

Option 5 is now selected to load the object code into the Evaluation Module for later execution by the TMS32010.

When this option is selected, the user is returned to the VMS operating system. The user types a 'control T' in order to toggle out of the transparency mode of the EVM. The EVM responds with its prompt of '?' and waits for further input.

% ~T (VMS prompt)

? lpm 2<cr> (TMS32010 EVM prompt)

which puts the EVM into a Load Program Memory mode. Note that no EVM prompt (?) is given after this command is given.

Another ~T is typed which puts the user back in touch with the VMS operating system. A VMS command of

~T

% type 3dfilt.mpo ~T

is given. Note that the command is followed by a ~T, NOT a carriage return. A carriage return will not work! The ~T puts the EVM back into local mode and sends a carriage return to the VMS system automatically. It is this carriage return which causes the VMS system to type out the file 3DFILT.MPO to the EVM.

Following the loading of the object file, a return to the VAX VMS system is required. This is accomplished by the following:

?~T <cr> <toggles back to VMS>

%r dices <executes DICES again>

The full main DICES menu is again presented.

6.6 PERFORMANCE EVALUATION

To evaluate the performance of the digital controller in a closed-loop system, option 6 is executed.

The current version of DICES can obtain the step response and frequency response of the closed-loop system. Each of these tests will be discussed in the following sections.

6.6.1 STEP RESPONSE

From the Performance Evaluation menu, select option 1 - Step Response Test. DICES will prompt for AMPLITUDE OF the STEP INPUT. Enter a carriage return for this value, since DICES defaults to a 1.0 volt input step. The next input is the APPROXIMATE SETTLING TIME for which 17 seconds is entered.

DICES then prompts to use a ~T to communicate with the EVM in order to start the digital filter program.

?PC 0000 <sets TMS32010 program counter to 0000>

?RUN <starts execution of the filter program>
 <communication is lost with DICES>

The START button in the CONTROL section on the pull-out drawer of the B/K 2032 is pressed to begin the system test (See figure 32).

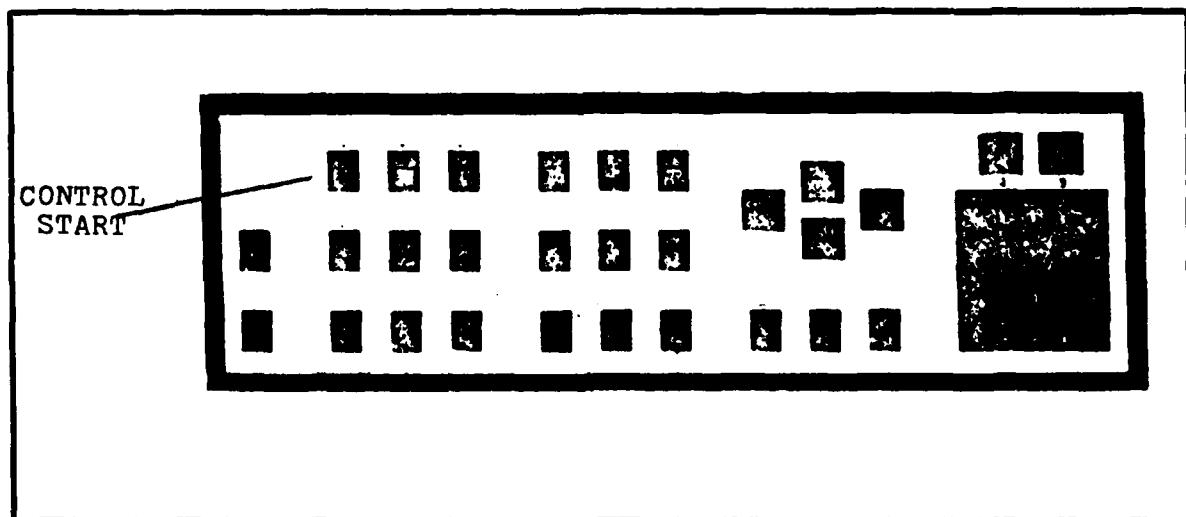


Figure 32. CONTROL-START Button on B/K 2032

The test will begin shortly after pressing the START button and will take approximately 32 seconds. After the test is complete, the B/K 2032 is placed into a mode that allows the user to move the cursor along the step response to determine the system peak time, settling time, peak value, etc. The value is read from the B/K 2032 screen in the upper right-hand corner of the screen.

Following the TEST COMPLETE message at the bottom of the B/K 2032, the RESET switch on the EVM must be asserted to stop the filter program and return to the EVM monitor program. From the monitor program, a ~T is typed which returns communication with DICES. A CARRIAGE RETURN is typed after the ~T and the DICES main menu is displayed.

With the compensator in the closed-loop, the analytical step response is shown in figure 33. As can be seen, the figures of merit are

$$t_p = 7 \text{ seconds}$$

$$M_p = 1.19 \text{ units}$$

$$t_s = 13.5 \text{ seconds}$$

The actual results of the test performed during this example are shown in figure 34. The figures of merit are

$$t_p = 6.5 \text{ seconds}$$

$$M_p = 1.2 \text{ units}$$

$$t_s = 11 \text{ seconds}$$

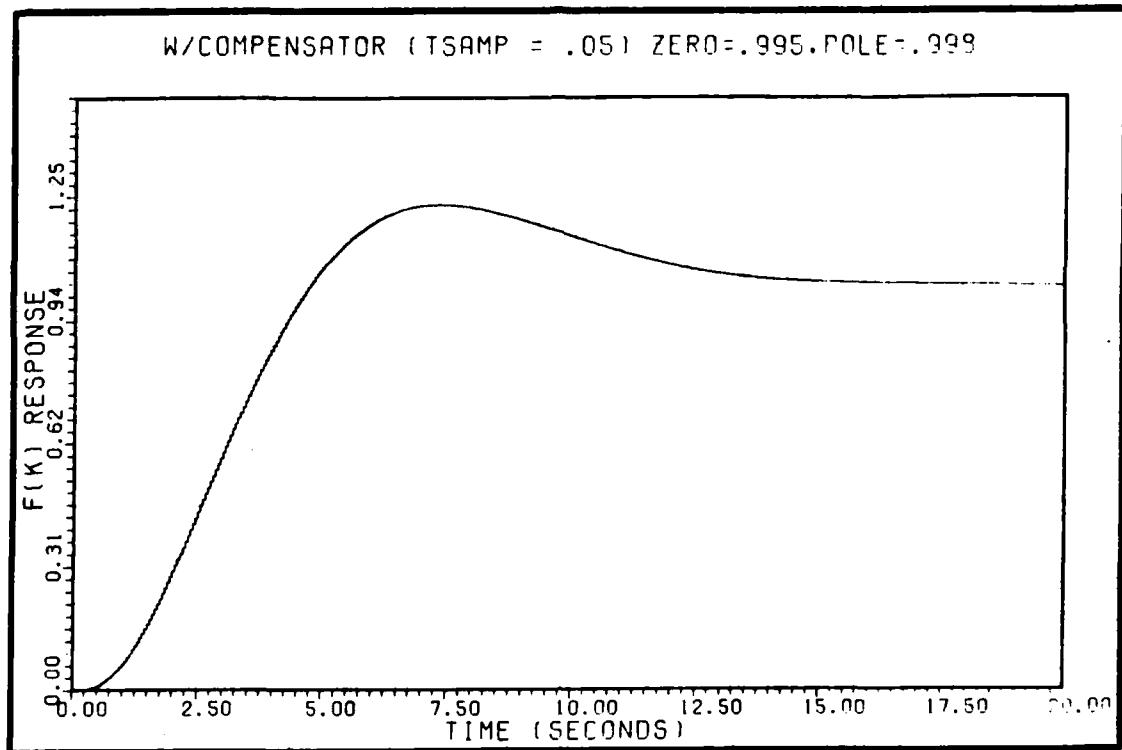


Figure 33. Compensated Theoretical Step Response

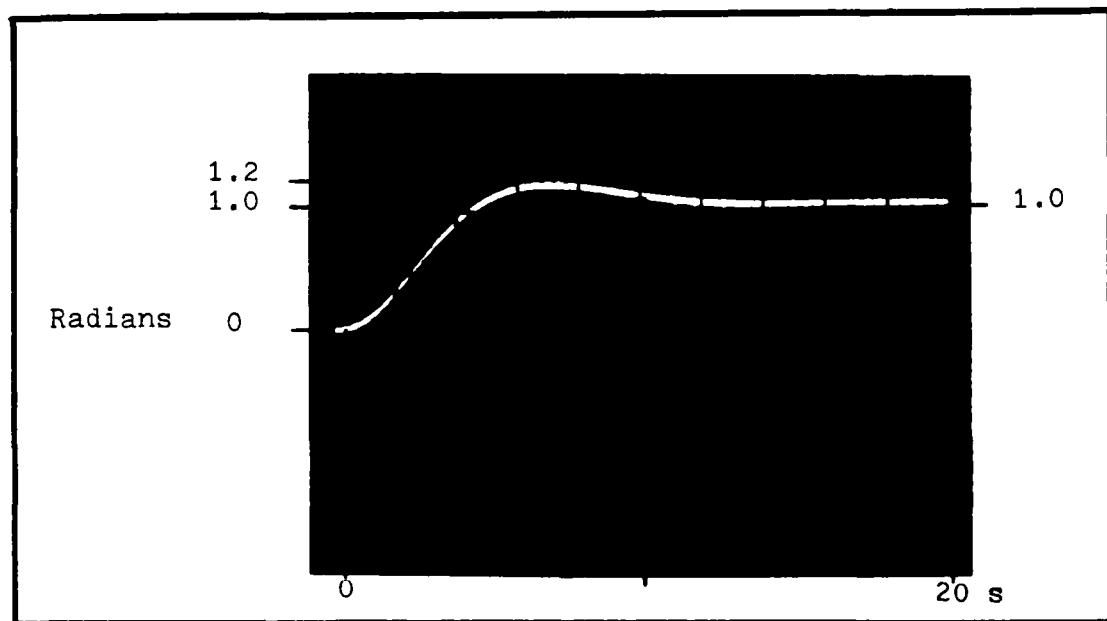


Figure 34. Actual Compensated Step Response

As can be seen, the performance due to an actual 16-bit digital filter implementation is roughly the same as the theoretical results obtained with ICECAP.

6.6.2 FREQUENCY RESPONSE

From the Performance Evaluation menu, select option 2 - FREQUENCY RESPONSE TEST. DICES will prompt for APPROXIMATE BANDWIDTH OF CLOSED-LOOP SYSTEM. Enter 10 for this value.

DICES then prompts to use a ~T to communicate with the EVM in order to start the digital filter program.

?PC 0000 <sets TMS32010 program counter to 0000>

?RUN <starts execution of the filter program>
<communication is lost with DICES>

The START button in the CONTROL section of the B/K 2032 is pressed to begin the system test (See figure 32).

The test will begin shortly after pressing the START button and will take approximately 7 MINUTES because of the low bandwidth of the system. This long test time is a consequence of the B/K method of performing the frequency response test of a system. After the test is complete, the B/K 2032 is placed into a mode that allows the user to move the cursor along the magnitude and phase curves simultaneously. The magnitude (in dB) and the phase of the plant output (in degrees) can be read from the right-hand

top and bottom of the screen, respectively.

Following the TEST COMPLETE message at the bottom of the B/K 2032, the RESET switch on the EVM must be asserted to stop the filter program and return to the EVM monitor program. From the monitor program, a ~T is typed which returns communication with DICES. A CARRIAGE RETURN is typed after the ~T and the DICES main menu is displayed.

6.7 REPORT GENERATION

The only documentation from the test currently available is a listing of the filter pairings that were performed and the coefficients generated during the quantization process. This report can be generated by selecting option 7 from the main menu.

6.8 SUMMARY

This section has shown how a typical control problem is taken from beginning to end using DICES. The user is free to change filter designs (within the current limitations) and make changes to the plant model if so desired. Each of the tests can be executed again if so desired.

7.0 KNOWN BUGS, ERRORS, AND TRICKS

This section describes known bugs, problems or

potential errors that may occur while using DICES.

Tag error

This error has occurred while downloading the 3DFILT.MPO object file from the VAX. It is detected and displayed by the EVM monitor program when the object data file is not received properly.

If this occurs, try the following:

1. Re-initialize the EVM as described in section 6.0
2. Insure the NOWRAP feature of the VT-100 is set.

B/K Interface I/O error

This error may occur during remote programming of the B/K 2032. It will be displayed on the bottom line of the B/K screen. Press the LOCAL key and then the STOP key on the DIGITAL I/O section of the B/K 2032.

GPIB-100 Initializing

If the VAX power is removed or the power is removed from the GPIB-100 units, it may be necessary to clear the units using the National Instruments supplied interactive program called IMICPX. This program allows remote programming the IEEE-488 instruments from the VT-100.

The following sequence is performed from directory
[ti.dices]:

```
%run imicpx          <executes imicpx>  
:g cl               <: is prompt from imicpx>  
                   <g cl clears the bus>  
:i cl 2             <clears B/K 2032>
```

DICES Will Not Program Instruments After VAX VMS Reboot

If DICES does not appear to program the test instruments on the IEEE-488 bus after the VAX VMS system has been rebooted, it may be necessary to re-connect the driver as described in the National Instruments GPIB-11-2 Software Reference Manual.

The driver connection should be made automatically because the [ti] log-in file has the CONNECT commands in it. But on one occasion, it did not do it automatically.

ICECAP will not Execute

If the error

FOR008.DAT not found (or something similar)
occurs, try the following:

```
%copy for008.bak for008.dat
```

Occasionally data file for008.dat gets wiped out by

ICECAP and it needs to be replaced from the backup that is kept in the directory.

Errors in Module Error Message From Assembler

If during the assembly of the source file 3DFILT.THS, a message 'Errors in Module' occurs, the reason is probably coefficients that are too large to implement as a 16-bit coefficient. This is caused by pole/zero combinations that generate a coefficient in the second-order difference equation that is greater than .999969. This may also be caused by too large a gain in the compensator.

8.0 REFERENCES

1. TMS32010 Evaluation Module User's Guide, SPRU005. Texas Instruments, Incorporated, February 1984.
2. TMS32010 Assembly Language Programmer's Guide, SPRU002B. Texas Instruments, Incorporated, November, 1983.
3. D'Azzo, John J. and Constantine H. Houpis. Linear Control Systems Analysis and Design. (Second Edition). New York: McGraw-Hill Book Company, 1981.
4. Houpis, Constantine H. and Gary B. Lamont. Digital Control Systems. New York: McGraw-Hill Book Company, 1985.
5. Gilbert, Richard. 'The General Purpose Interface Bus,' IEEE Micro: 41-51 (February 1982).
6. Phillips, Charles H., and H. Troy Nagle, Jr. Digital Control System Analysis and Design. Englewood Cliffs, NJ: Prentice Hall, Inc, 1984.
7. Larimer, Stanley J. An Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS Thesis. Wright Patterson AFB, OH: Air Force Institute of Technology, March 1978.

8. Logan, Glen T. Development of an Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS Thesis. Wright Patterson AFB, OH: Air Force Institute of Technology, March 1982.
9. Carlson, Alan, George Harnauer, Thomas Carey, and Peter J. Holsberg (Editors). Handbook of Analog Computation. (Second Edition). Princeton, New Jersey: Electronic Associates, Inc, 1967.
10. ICECAP User's Manual. No Date.
11. TMS32010 Analog Interface Board Data Manual. (Release 2). Texas Instruments, Incorporated, 1983.
12. IEEE-488 Standard Digital Interface for Programmable Instruments. IEEE, New York, 1978.
13. GPIB11-2 Hardware Manual. National Instruments, Incorporated, March, 1985.
14. GPIB11 Series Multiboard Driver VAX VMS Software Reference Manual. P/N 310005-04. National Instruments, Incorporated, March, 1985.
15. Analog-to-Digital Convertors. Volume 1. 10-83. Specification Sheet for ADC80. Analog Devices.

16. Instruction Manual for the Wavetek 172B Programmable Signal Source. Wavetek, October, 1979.
17. B/K 2032 Instruction Manual IEC/IEEE Interface (Vol. 3). Brueel and Kjaer. September, 1983.
18. Clune, Thomas R. 'Interfacing for Data Acquisition,' Byte Magazine, Vol 10, pp. 269-282, February, 1985.
19. B/K 2032 Instruction Manual - Operation (Vol. 2). Brueel and Kjaer. September, 1983.

Attachment 1
TMS32010 EVM Unit Description

Introduction

This appendix describes the TMS32010 EVM Unit that is used as the digital controller in DICES. The EVM Unit contains a 16-bit TMS32010 Digital Signal Processor microprocessor and a Analog-to-Digital (A/D) and Digital-to-Analog (D/A) converter.

Description of EVM

The TMS32010 is a Digital-Signal-Processing (DSP) microprocessor introduced by Texas Instruments, Inc. in 1983. Its architecture, speed, and instruction set are designed for efficient execution of digital signal processing algorithms. The table below summarizes its main features and performance characteristics.

Configuration: Microprocessor version - TMS32010.

Clock/Instruction Cycle:	20 Mhz clock/200 nsec instruction cycle. On-chip oscillator.
Memory:	144 Words of on-chip data RAM. 4096 words of external program ROM.
Computational Sections:	Double-Precision 32-bit ALU and Accum. 200 nsec multiplier. Overflow mode that saturates the accumulator on arithmetic overflow.
Program Control Section:	16-bit bus for off-chip instructions. 4x12 stack for context switching. Autoincrement/decrement registers. Polling pin for hardware interfacing. Vectored interrupt.

The TMS32010 uses a modified Harvard architecture, (see figure 34), which permits the transfer of information between program and data memories. A strict Harvard architecture provides separate program and data memories with no cross transfer of data. This modified structure allows the designer the flexibility, for example, to reload the filter coefficients and the initial conditions of his control algorithm.

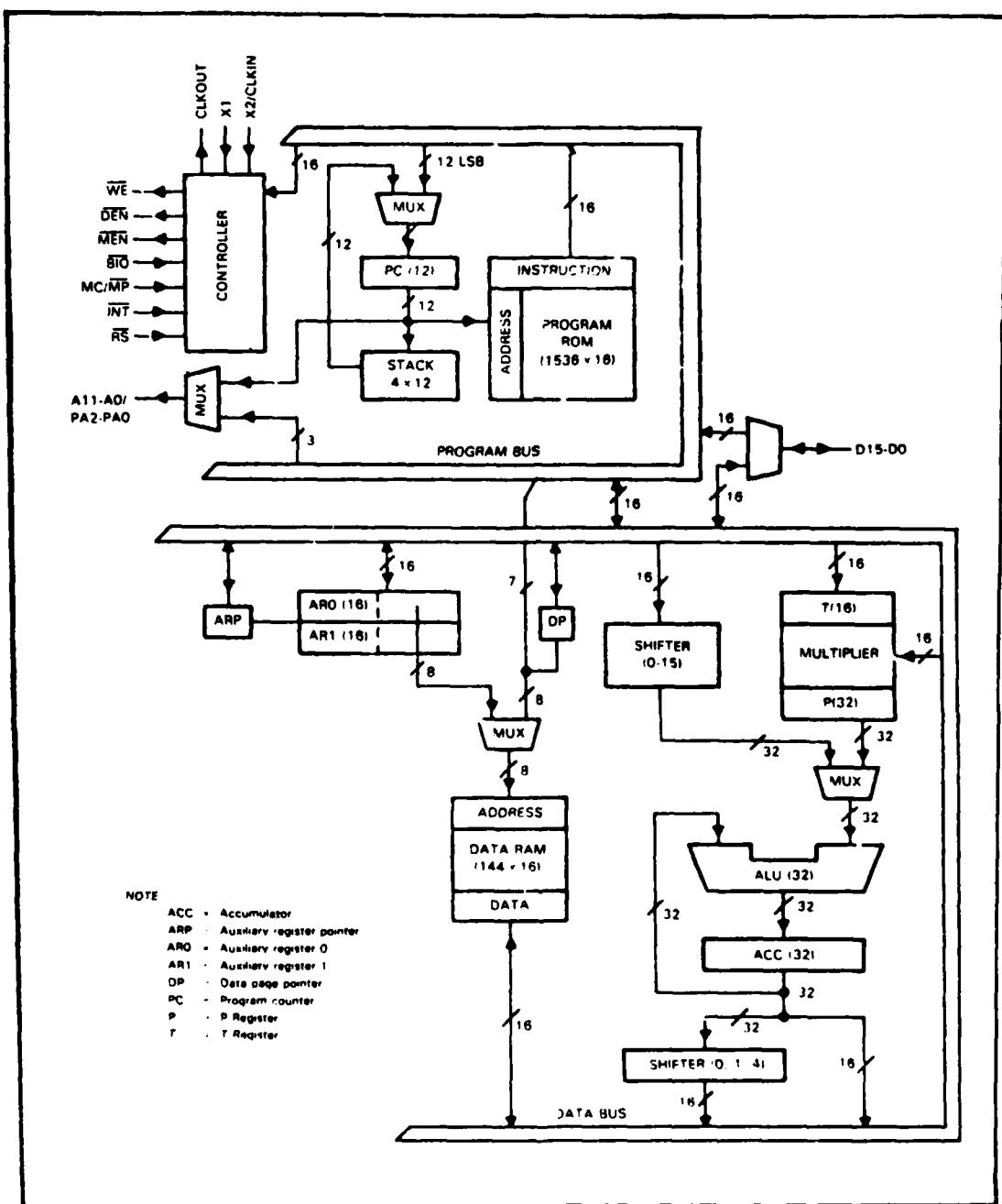


Figure 34. Modified Harvard Architecture

An outstanding feature is an on-board 16x16 bit multiplier, which produces a 32-bit product in one instruction cycle, i.e., 200 nsec. The TMS32010 also contains a barrel shifter that left shifts data 0 to 15 places before the data is transferred into the accumulator and a second shifter that shifts the upper half of the 32-bit accumulator 0, 1, or 4 places before it is stored in data RAM. This allows for the extraneous bits that are generated when using Q15 and Q12 number representation within the TMS32010. The I/O structure consists of eight 16-bit input and eight 16-bit output ports.

The TMS32010 instruction set includes a set of special instructions necessary for fast implementation of sum-of-products computations encountered in digital filtering/compensation calculations. Most of the instructions critical to DSP execute in 200 nsec. A typical multiplication of a delayed value by a coefficient stored in memory, and then a shift of the delayed data in memory executes in 400 nsec. The results of the previous operation is also accumulated in the accumulator during the same 400 nsecs. The TMS32010 has three addressing modes: direct, indirect, and immediate addressing.

Appendix B - Data Dictionary

Contents

	Page
1.0 Introduction	298
2.0 Data Element/Data Flow Descriptions	299
3.0 Process Descriptions	305
4.0 File Descriptions	312
5.0 Subroutines	314

Section 1. Introduction

This appendix contains the data dictionary for the Digital Interactive Controller Evaluation System (DICES). The data dictionary consists of four sections.

Section 2.0 describes the data elements and data flows to assist in using the data flow diagrams in the main text. However, this section can be read without the diagrams.

Section 3.0 provides the process descriptions. Again, this section is most helpful when used with the data flow diagrams, but it is not mandatory to have the diagrams.

Section 4.0 describes each file used by DICES. These files include data, text, and object files.

Finally, Section 5.0 describes each subroutine called by the main module DICES.THS and any routines each called module makes.

Section 2.0 Data Element/Data Flow Descriptions

DATA ELEMENT NAME: Perf_Reqts

DESCRIPTION: Performance requirements that have been established by the user.

FROM PROCESS: 1.0

DATA ELEMENT NAME: Math_Model

DESCRIPTION: Mathematical model of the plant under to be compensated.

FROM PROCESS: 2.0

DATA ELEMENT NAME: Plant_Sim

DESCRIPTION: Plant simulation on analog computer.

FROM PROCESS: 3.0

DATA ELEMENT NAME: Design_Parameters

DESCRIPTION: Parameters that describe the digital controller that has been designed using ICECAP CAD package.

FROM PROCESS: 4.0

DATA ELEMENT NAME: Implementation_Parameters

DESCRIPTION: Parameters that describe the actual coefficients that have been quantized and converted to integers to be implemented on the digital controller.

FROM PROCESS: 5.0

DATA ELEMENT NAME: Instructions

DESCRIPTION: Instructions to user on system configuration.

FROM PROCESS: 3.1

DATA ELEMENT NAME: Wired_Sim

DESCRIPTION: A wired analog computer that simulates the plant.

FROM PROCESS: 3.2

DATA ELEMENT NAME: Working_Sim

DESCRIPTION: A wired analog computer that accurately simulates the plant.

FROM PROCESS: 3.3

DATA ELEMENT NAME: Sys_Config

DESCRIPTION: A configured system that has the digital controller connected to the plant simulation in a closed-loop.

FROM PROCESS: 3.4

DATA ELEMENT NAME: Model_Specs

DESCRIPTION: Mathematical model of the plant and performance specifications.

FROM PROCESS: 4.1

DATA ELEMENT NAME: Design_Parameters

DESCRIPTION: Parameters that describe the compensator that has been design to meet the performance specifications.

FROM PROCESS: 4.22 or 4.23

DATA ELEMENT NAME: Verified_Compensator

DESCRIPTION: Verified compensator that meets the performance specifications analytically.

FROM PROCESS: 4.3

DATA ELEMENT NAME: Coefficient_File

DESCRIPTION: Data file that contains the parameters of the compensator.

FROM PROCESS: 4.4

DATA ELEMENT NAME: Pole/Zero_Array

DESCRIPTION: Array of poles and zeros of the compensator.

FROM PROCESS: 5.1

DATA ELEMENT NAME: Implementation_Structure

DESCRIPTION: Filter structure type that will be used to implement the digital controller.

FROM PROCESS: 5.2

DATA ELEMENT NAME: Filter_Sections

DESCRIPTION: Paired sets of poles and zeros that will form the first or second order sections of the digital controller.

FROM PROCESS: 5.3

DATA ELEMENT NAME: Quantized_Coefficients

DESCRIPTION: Quantized numerator and denominator coefficients using a 16-bit integer representation. The sample period constant is also passed as a scaled integer.

FROM PROCESS: 5.4

DATA ELEMENT NAME: Possible_New_Design

DESCRIPTION: If the analysis shows that performance specifications have not been met, a new design is accomplished and passed to process 5.1.

FROM PROCESS: 5.5

DATA ELEMENT NAME: Scaled_Coefficients

DESCRIPTION: Scaled versions of the controller coefficients to reduce the probability of arithmetic overflow.

FROM PROCESS: 5.6

DATA ELEMENT NAME: Sample_period

DESCRIPTION: Integer number that represents the sample period to be used on the TMS32010 EVM board. This is a scaled number to account for the limited range of sample times available on the board.

FROM PROCESS: 5.7

DATA ELEMENT NAME: Object_Code

DESCRIPTION: TMS32010 object code that when executed will implement the digital controller.

FROM PROCESS: 5.8

DATA ELEMENT NAME: Digital_Controller

DESCRIPTION: Digital controller that is ready to execute the controller algorithm.

FROM PROCESS: 5.9

DATA ELEMENT NAME: Test_Requirements

DESCRIPTION: Test requirements as specified by the user.

FROM PROCESS: 6.1

DATA ELEMENT NAME: Command_String

DESCRIPTION: ASCII command string that represents the commands to be sent to the programmable test equipment to set up the proper functions, ranges, etc.

FROM PROCESS: 6.2

DATA ELEMENT NAME: Programmed_Test_Equipment

DESCRIPTION: After receipt of the command string, the test equipment is properly configured for tests to be performed.

FROM PROCESS: 6.3

DATA ELEMENT NAME: Test_Sequence

DESCRIPTION: Sequence of test operations performed by
the programmable test equipment.

FROM PROCESS: 6.4

DATA ELEMENT NAME: Test_Data

DESCRIPTION: Results from tests performed by test
equipment.

FROM PROCESS: 6.5

Section 3.0 Process Descriptions

PROCESS NO: 1.0

PROCESS NAME: Establish System Performance Specifications

DESCRIPTION: Establishes system performance specifications for the system under investigation. Performed by the user external to DICES.

ALIASES:

PROCESS NO: 2.0

PROCESS NAME: Develop Plant Model

DESCRIPTION: Develop a mathematical model for the plant under investigation which is later used as the basis for the controller design.

ALIASES:

PROCESS NO: 3.0

PROCESS NAME: Simulate Plant Model

DESCRIPTION: Using conventional analog computer techniques, the plant model is simulated for use during the hybrid simulation..

ALIASES:

PROCESS NO: 3.1

PROCESS NAME: Display Simulation Options

DESCRIPTION: Presents user with various options to assist in the simulation process.

ALIASES:

PROCESS NO: 3.2

PROCESS NAME: Patch-Up Equations

DESCRIPTION: Physical process of actually wiring the analog computer circuits to implement the plant model.

ALIASES:

PROCESS NO: 3.3

PROCESS NAME: Test Simulation Model

DESCRIPTION: Stimulation of the closed or open-loop system to determine how well the model matches analytical performance obtained by ICECAP or TOTAL.

ALIASES:

PROCESS NO: 3.4

PROCESS NAME: Connect Controller

DESCRIPTION: Physically inserting the digital controller into either the forward or feedback path of the simulation.

ALIASES:

PROCESS NO: 4.0

PROCESS NAME: Design Controller

DESCRIPTION: Obtain the transfer function of a digital controller which will cause the closed-loop system to perform within the established performance specifications.

ALIASES:

PROCESS NO: 4.1

PROCESS NAME: Access CAD Package

DESCRIPTION: Connect the user to a CAD package (ICECAP) to allow design of the controller.

ALIASES:

PROCESS NO: 4.2

PROCESS NAME: Standard Design Techniques

DESCRIPTION: Using standard design techniques, obtain the controller transfer function which will cause the system to perform within specifications.

ALIASES:

PROCESS NO: 4.3

PROCESS NAME: Analyze System Performance

DESCRIPTION: Using the controller design, obtain the analytical performance characteristics of the closed-loop system and compare with the desired specifications.

ALIASES:

PROCESS NO: 4.4

PROCESS NAME: Save Controller Coefficients

DESCRIPTION: Stores the coefficients of the controller to allow later access by DICES.

ALIASES:

PROCESS NO: 5.0

PROCESS NAME: Implement Controller

DESCRIPTION: Using the analytical design of the digital controller, implement with a filter structure of cascade second-order filter sections.

ALIASES:

PROCESS NO: 5.1

PROCESS NAME: Retrieve Controller Factors

DESCRIPTION: Retrieve the stored factors for the controller from memory to be used in the filter implementation process.

ALIASES:

PROCESS NO: 5.2

PROCESS NAME: Pick Filter Structure

DESCRIPTION: Select the structure of the filter desired to implement the second-order sections. (1D, 2D, 3D, 4D, 1X, or 2X).

ALIASES:

PROCESS NO: 5.3

PROCESS NAME: Pole-Zero Pairing

DESCRIPTION: Pairs the controller poles and zeros using an algorithm to minimize the effects of coefficient quantization.

ALIASES:

PROCESS NO: 5.4

PROCESS NAME: Quantize Coefficients

DESCRIPTION: Quantizes the coefficients of the controller into 16-bit digital values for implementation on the TMS32010 microprocessor.

ALIASES:

PROCESS NO: 5.5

PROCESS NAME: Re-Analyze Performance

DESCRIPTION: Using the 'new' quantized coefficients, re-analyze the design using ICECAP.

ALIASES:

PROCESS NO: 5.6

PROCESS NAME: Scale Factor and Ordering

DESCRIPTION: Determines the proper scale factor for the controller algorithm to avoid overflow. Orders the second-order sections to minimize the effects of round-off and coefficient quantization noise.

ALIASES:

PROCESS NO: 5.7

PROCESS NAME: Sample-Period Selection

DESCRIPTION: Selects sample-period to be used on the TMS32010 EVM board to sample the input signal.

ALIASES:

PROCESS NO: 5.8

PROCESS NAME: Generate Code

DESCRIPTION: Loads the quantized coefficients and sample-period into the TMS32010 digital filter source file. The source file is then assembled into object code using the TMS32010 Assembler

ALIASES:

PROCESS NO: 5.9

PROCESS NAME: Load TMS32010 Object Code

DESCRIPTION: Loads the TMS32010 object code into the program memory of the TMS32010 EVM for execution.

ALIASES:

PROCESS NO: 6.1

PROCESS NAME: Obtain Test Requirements

DESCRIPTION: Obtains the test requirements from the user for the system-under-test.

ALIASES:

PROCESS NO: 6.2

PROCESS NAME: Build Command String

DESCRIPTION: Assembles the required ASCII character string that represents the set of commands to the various test equipment.

ALIASES:

PROCESS NO: 6.3

PROCESS NAME: Send Command String

DESCRIPTION: Sends the command string to each piece of test equipment that requires remote programming to perform a particular test.

ALIASES:

PROCESS NO: 6.4

PROCESS NAME: Initiate Test

DESCRIPTION: Starts the test sequence.

ALIASES:

PROCESS NO: 6.5

PROCESS NAME: Receive Test Results

DESCRIPTION: Collects results from the various tests that have been performed on the system-under-test.

ALIASES:

Section 4.0 File Descriptions

FILE NAME: 3DFILT.LIS

DESCRIPTION: List file generated from assembly of
3DFILT.THS

PROCESS USED: 5.8

ACCESS: N/A

FILE NAME: 3DFILT.MPO

DESCRIPTION: TMS32010 Object code from assembly of
3DFILT.THS. The format is TMS9900.

PROCESS USED: 5.8

ACCESS: Sequential

FILE NAME: 3DFILT.THS

DESCRIPTION: Generic eighth-order 3D filter for TMS32010
written in TMS320 Macro Assembler.

PROCESS USED: 5.8

ACCESS: Sequential

FILE NAME: MEMORY.DAT

DESCRIPTION: Contains all the design parameters from a
session with ICECAP.

PROCESS USED: 5.1

ACCESS: Sequential

FILE NAME: NEW3D.ECK

DESCRIPTION: A temporary file that is used to store 3DFILT.THS as its coefficients are changed and sample rate inserted into the program. This file is transferred back to 3DFILT.THS when complete and NEW3D.ECK is deleted.

PROCESS USED: 5.2

ACCESS: Sequential

FILE NAME: NEWCO.TXT

DESCRIPTION: A temporary file used to store the quantized integer coefficients and sample period constant.

PROCESS USED: 5.8

ACCESS: Sequential

FILE NAME: PAIR.PRT

DESCRIPTION: A print file used to collect output from the pole/zero pairing process and the quantization process.

PROCESS USED: 5.3 and 5.4

ACCESS: Sequential

Section 5.0 Subroutines

MODULE NAME: ANYCOMPZEROS

DESCRIPTION: Determines if any complex zeros are available (exist or not used in the pairing process yet).

CALLS: None

MODULE NAME: ANYREALZEROS

DESCRIPTION: Determines if any real zeros are available (exist or not used yet in the pairing process).

CALLS: None

MODULE NAME: CALCFREQSPAN

DESCRIPTION: Calculated proper frequency span setting for the B/K 2032 in order to get the proper time base for step response tests.

CALLS: None

MODULE NAME: CHARTONUM

DESCRIPTION: Converts a single input character to a single integer.

CALLS: None

MODULE NAME: CHGCO

DESCRIPTION: Reads an ASCII file program file, modifies lines 168-188, and then rewrites the file back under original name. Deletes the temporary file that is used.

CALLS: None

MODULE NAME: CLEARSCREEN

DESCRIPTION: Clears the CRT screen.

CALLS: None

MODULE NAME: CLOSESTZERO

DESCRIPTION: Finds closest single pole to a given reference pole.

CALLS: None

MODULE NAME: CLOSESTREALPAIRC

DESCRIPTION: Finds the closest real pair of zeros to a reference pole (assumes that the reference pole is complex).

CALLS: None

MODULE NAME: CLOSESTCOMPLEXPAIR

DESCRIPTION: Finds closest complex pair of zeros to reference pole (assumes that two or more zeros are available).

CALLS: None

MODULE NAME: CLOSESTREALPAIRR

DESCRIPTION: Finds closest real pair of zeros to reference pole (assumes reference pole is real).

CALLS: None

MODULE NAME: CONFIGEQUIP

DESCRIPTION: Determines if test equipment is turned on and configures equipment to initial default settings if on. If not on, it simply returns. Prints a 'DICES' message on the B/K if turned on.

CALLS: INITCONFIG

MODULE NAME: CONVCHAR

DESCRIPTION: Converts a real number between 0 and 10 to a 3-character string, e.g., 2.3454 is converted to '2.3'.

CALLS: None

MODULE NAME: CONVCHARFS

DESCRIPTION: Converts a real number between 0 and 25600 to a 5-character string with no decimal point. e.g., 232.3454 is converted to '232 '.

CALLS: None

MODULE NAME: DICES

DESCRIPTION: Main module that coordinates all the calls to all the other subroutines.

CALLS: CLEARSCREEN, RDORDER, RDHTF, PAIR, WAIT,
MULTRTS

MODULE NAME: FDV\$CDISP

DESCRIPTION: Call to Forms Management System (FMS) to display a particular form from the declared library.

CALLS: None

MODULE NAME: FDV\$GET

DESCRIPTION: Call to Forms Management System (FMS) to get a field value.

CALLS: None

MODULE NAME: FDV\$GETAL

DESCRIPTION: Call to Forms Management System (FMS) to get all the field values from a particular form.

CALLS: None

MODULE NAME: FDV\$INIT

DESCRIPTION: Initializes VMS to use the Forms Management System.

CALLS: None

MODULE NAME: FDV\$LCHAN

DESCRIPTION: Opens a channel in VMS to use FMS.

CALLS: None

MODULE NAME: FDV\$LOPEN

DESCRIPTION: Opens user form library for Forms Management System use.

CALLS: None

MODULE NAME: INITCONFIG

DESCRIPTION: Initializes the B/K 2032 and Wavetek 172B to default settings. Stores the set-ups in the B/K for later use by DICES.

CALLS: WRITEMSG

MODULE NAME: MARK1Z

DESCRIPTION: Marks one real zero in vector that contains index numbers of zeros that are available for pairing.

CALLS: None

MODULE NAME: MARK2Z

DESCRIPTION: Marks two real zeros in vector that contains index numbers of zeros that are available for pairing.

CALLS: None

MODULE NAME: MULTRTS

DESCRIPTION: Multiplies out the roots of the numerator and denominator of each second-order section to yield first or second order polynomials. The coefficients are then quantized and converted to a 16-bit integer representation for execution on the TMS32010.

CALLS: None

MODULE NAME: PAIR

DESCRIPTION: Pairs the poles and zeros of the controller into first or second order sections using an algorithm described in Phillips and Nagle (13).

CALLS: MARK1Z
MARK2Z
ANYCOMPZEROS
ANYREALZEROS
CLOSEST1ZERO
CLOSESTREALPAIRR
CLOSESTREALPAIRC
CLOSESTCOMPLEXPAIR

MODULE NAME: PAIRPRT

DESCRIPTION: Prints the file PAIR.PRT on the line printer.

CALLS: None

MODULE NAME: RDHTF

DESCRIPTION: Reads data file MEMORY.DAT to extract the zeros and poles of the compensator as designed using ICECAP.

CALLS: None

MODULE NAME: RDORDER

DESCRIPTION: Reads the order of the numerator and denominator, the loop sensitivity (HK), and the sample period of the compensator from MEMORY.DAT.

CALLS: None

MODULE NAME: SETBKINLEVEL

DESCRIPTION: Programs the proper maximum input level value into the B/K.

CALLS: WRITEMSG
CONVCHAR

MODULE NAME: SIXCHAR

DESCRIPTION: Converts six character digits to an integer number.

CALLS: None

MODULE NAME: THREECHAR

DESCRIPTION: Converts three character digits to an integer.

CALLS: None

MODULE NAME: TWOCHAR

DESCRIPTION: Converts two character digits to an integer.

CALLS: None

MODULE NAME: WAIT

DESCRIPTION: Displays a message to hit RETURN when ready to continue and waits for the RETURN to be entered.

CALLS: None

MODULE NAME: WAITSTART

DESCRIPTION: Waits for the pressing of the 'START' button on the B/K 2032 (CONTROL). Used to signal DICES that user is ready to initiate test. Moves B/K field selector to cursor position to allow user to determine frequency, time, and/or amplitude values from display.

CALLS: WRITEMSG

MODULE NAME: WRITEMSG

DESCRIPTION: Takes any length character string and adds a Linefeed character to the end. This character is used as an IEEE-488 terminator character to signal the end of the message. This routine also determines length of message and adds one because of the LF character. Routine receives the device number of device to receive message. Calls the IBUPU function which sends the message to the specified device.

CALLS: WRITEMSG
IBUPU (function)

Appendix C - Data Flow Diagram

Data Flow Diagrams

The following is a very brief description of data flow diagrams taken from Houpis and Lamont (6:427-433).

Data Flow Diagrams (DFDs) are used to portray processes and their interrelationships (see figure C-1). Various levels of diagrams are used to build a hierarchy of processes. This hierarchy permits system overview at the high levels and process details at the low levels.

A DFD is a graphical representation of a system emphasizing the data flow of the overall process. The important aspect of the DFD is its ability (if it used properly) to represent the component processes of the system and also their respective interfaces. Many logical errors can occur in system software design if the information being transferred from process to process is not specified properly.

In generating a DFD, it is best to focus first on the input and output data flows in order to establish the DFD context. Next, the additional circles (processes) are added to connect the input and output, resulting in an overall data flow. As the circles are added, it is necessary to try to define processes as activities at the same level (i.e., manipulating the same level of data structures, or the same level of actions).

In defining the processes as well as the data items, labels are used that represent the physical reality of the

overall system. A check is made to ensure that the interface data flows are correct. Each circle should have a strong single-action verb and data object. The objective of understandability dictates these constraints.

To provide the capability of representing complex and sophisticated systems, a diagram hierarchy is developed as shown in figure C-1. Each of the circles (processes) can be decomposed into lower-level diagrams that describe in more detail the actual process. For example, the compensator calculation in figure C-1 could be decomposed as shown in figure C-2. Each lower-level diagram is then contained within a context itself, i.e., a parent-child relationship.

The same circle-level concept can be applied to the arrows, i.e., at the high levels data are grouped (pipelined) and at the lower levels they are decomposed into specific data items. A maximum of seven circles per diagram (one page) is suggested for maximum understandability.

Continuing refinement is required as the DFDs are being developed. Analysis should focus on the search for missing data flows, extra data flows, and correct decomposition. Peer review can be very helpful in this regard prior to the initiation of the design phase. The DFD should eventually be configuration controlled, i.e., modifications of requirements have to be reviewed and approved before incorporation.

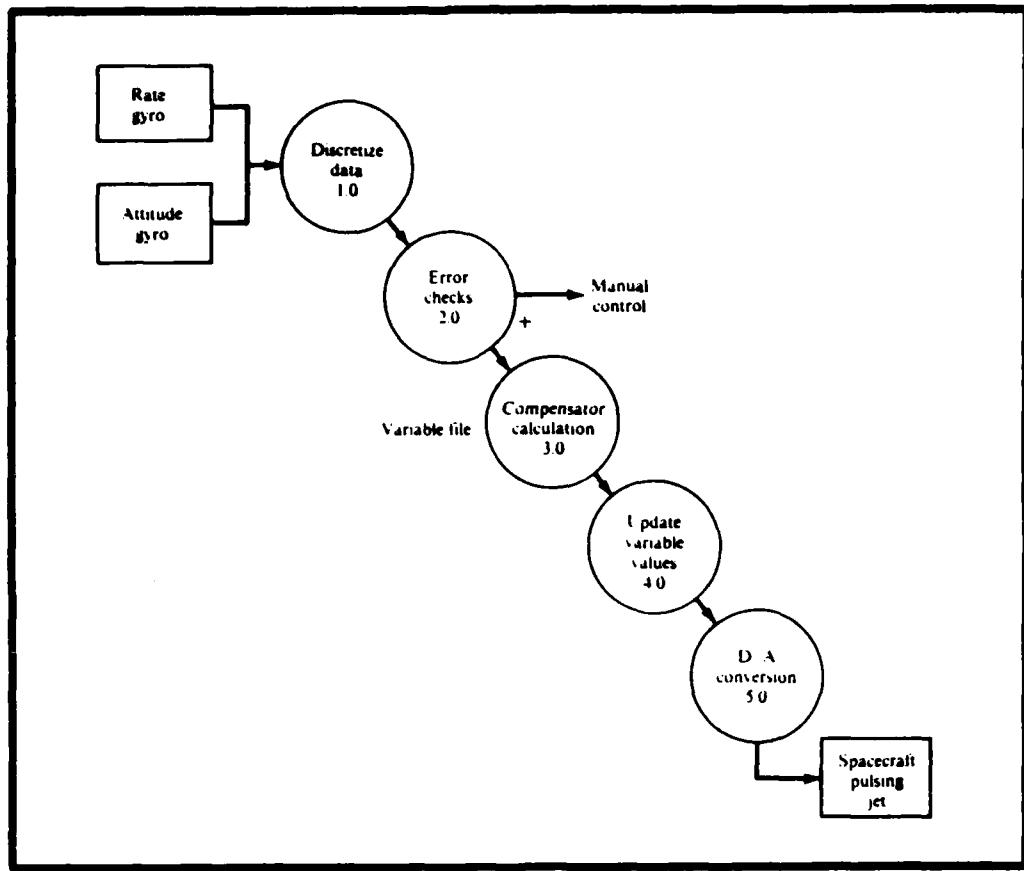


Figure C-1. Data Flow Diagram

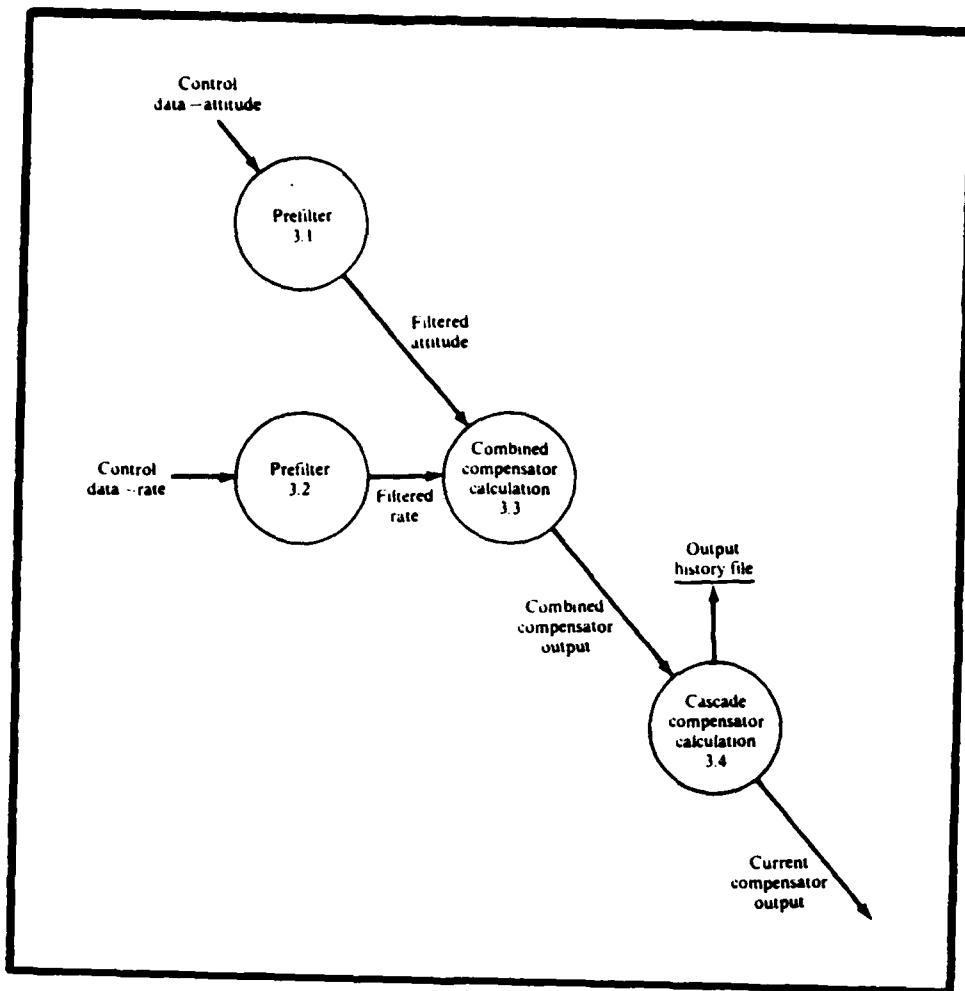


Figure C-2. Compensator Calculation (DFD3.0)

Appendix D - Test Data

Table of Contents

POLE/ZERO PAIRING TEST	329
TEST CASES	332
READ HTF MODULE TEST	351
MULTIPLY ROOTS TEST	354
NEW COEFFICIENTS TEST	357

Module: PAIR.THS

Pole-Zero Pairing Test

This test was designed to test the pole-zero pairing algorithm implemented in PAIR.THS. The test input data consisted of 12 combinations of poles and zeros.

Each test combination was chosen to test a particular path through the algorithm flowchart. For each combination, the path taken through the algorithm flowchart is shown along with the pole-zero map.

Each test case pole-zero map also shows the algorithm's pairing of the poles and zeros (which were correct in every case). This is shown by dotted lines around the pole-zero pairs.

Each test data print-out shows the number of sections required to form the filter, the poles and zeros of the filter, and the pole-zero pairs for each of the required sections.

An example of the computer printout of the test data and path through the algorithm follows on the next page for test case 2B.

Test Summary. All the pole-zero pairings were executed as expected. The module appears to meet all the requirements placed upon it.

Example of Test Data Print-out:

TEST CASE 2B

OF SECTIONS 3 <---- # of sections required

ZEROS

(0.3000000,0.1000000)
(0.3000000,-0.1000000)
(0.5000000,0.2000000) <---- Unpaired zeros of filter
(0.5000000,-0.2000000)
(0.5000000,0.0000000E+00)

POLES

(0.5000000,0.3000000)
(0.5000000,-0.3000000)
(0.3000000,0.2000000) <---- Unpaired poles of filter
(0.3000000,-0.2000000)
(0.6000000,0.0000000E+00)

SECTION NUMBER = 1

ZEROS(1) = (0.5000000,0.0000000E+00) <--- Sect #1 zeros
ZEROS(2) = (9999.000,9999.000)
POLES(1) = (0.6000000,0.0000000E+00) <--- Sect #1 poles
POLES(2) = (9999.000,9999.000)

SECTION NUMBER = 2

ZEROS(1) = (0.5000000,0.2000000) <--- Sect #2 zeros
ZEROS(2) = (0.5000000,-0.2000000)
POLES(1) = (0.5000000,0.3000000) <--- Sect #2 poles
POLES(2) = (0.5000000,-0.3000000)

SECTION NUMBER = 3

ZEROS(1) = (0.3000000,0.1000000) <--- Sect #3 zeros
ZEROS(2) = (0.3000000,-0.1000000)
POLES(1) = (0.3000000,0.2000000) <--- Sect #3 poles
POLES(2) = (0.3000000,-0.2000000)

Note: 9999.000 represents unused poles and zeros

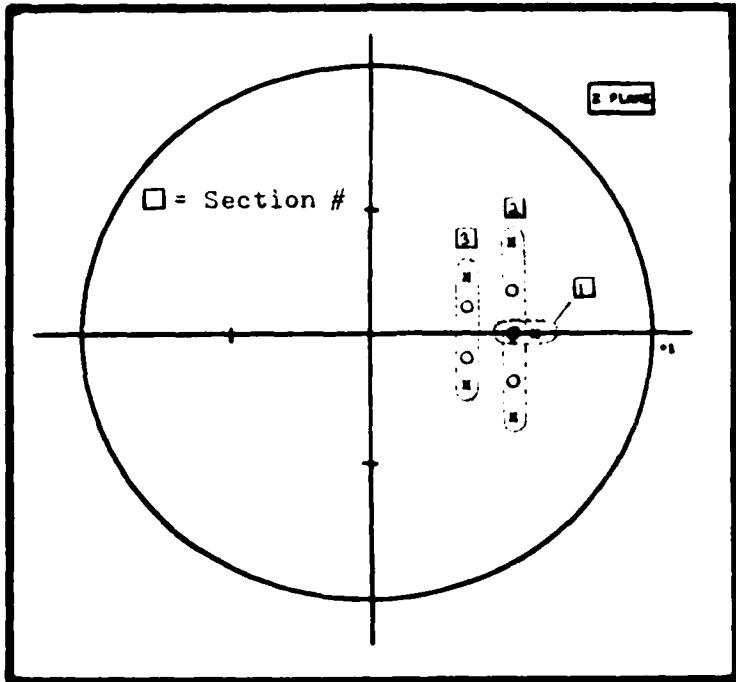
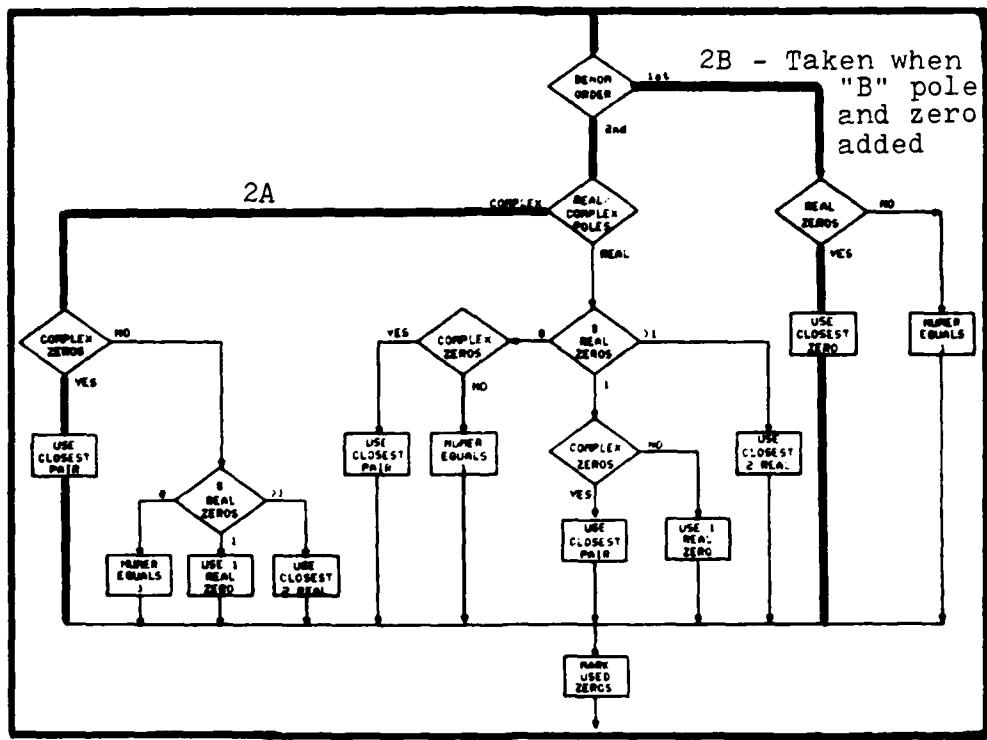


Figure D-1. Example Pole-Zero Map and Pairing Algorithm

Case A tests consisted of an even number of poles and combinations of zeros.

Case B consisted of all Case A poles and zeros plus 1 real pole and zero.

The test results are shown in figures D-2 through D-7.

Case A

	<u>Poles</u>	<u>Zeros</u>
1A	Complex	Real
2A	Complex	Complex
3A	Complex	Real and Complex
4A	Real	Real
5A	Real	Complex
6A	Real	Real and Complex

Case B

	<u>Poles</u>	<u>Zeros</u>
1B	Complex and Real	Real
2B	Complex and Real	Complex
3B	Complex and Real	Real and Complex
4B	Real	Real
5B	Real	Real and Complex
6B	Real	Real and Complex

TEST CASE 1A

OF SECTIONS 2

ZEROS

(0.2000000,0.0000000E+00)
(0.3000000,0.0000000E+00)
(0.8000000,0.0000000E+00)
(0.9000000,0.0000000E+00)

POLES

(0.5000000,0.3000000)
(0.5000000,-0.3000000)
(0.3000000,0.2000000)
(0.3000000,-0.2000000)

SECTION NUMBER = 1

ZEROS(1) = (0.3000000,0.0000000E+00)
ZEROS(2) = (0.2000000,0.0000000E+00)
POLES(1) = (0.5000000,0.3000000)
POLES(2) = (0.5000000,-0.3000000)

SECTION NUMBER = 2

ZEROS(1) = (0.8000000,0.0000000E+00)
ZEROS(2) = (0.9000000,0.0000000E+00)
POLES(1) = (0.3000000,0.2000000)
POLES(2) = (0.3000000,-0.2000000)

TEST CASE 1B

OF SECTIONS 3

ZEROS

(0.2000000,0.0000000E+00)	POLES
(0.3000000,0.0000000E+00)	(0.5000000,0.3000000)
(0.8000000,0.0000000E+00)	(0.5000000,-0.3000000)
(0.9000000,0.0000000E+00)	(0.3000000,0.2000000)
(0.5000000,0.0000000E+00)	(0.3000000,-0.2000000)
	(0.6000000,0.0000000E+00)

SECTION NUMBER = 1

ZEROS(1) = (0.5000000,0.0000000E+00)
ZEROS(2) = (9999.000,9999.000)
POLES(1) = (0.6000000,0.0000000E+00)
POLES(2) = (9999.000,9999.000)

SECTION NUMBER = 2

ZEROS(1) = (0.3000000,0.0000000E+00)
ZEROS(2) = (0.2000000,0.0000000E+00)
POLES(1) = (0.5000000,0.3000000)
POLES(2) = (0.5000000,-0.3000000)

SECTION NUMBER = 3

ZEROS(1) = (0.8000000,0.0000000E+00)
ZEROS(2) = (0.9000000,0.0000000E+00)
POLES(1) = (0.3000000,0.2000000)
POLES(2) = (0.3000000,-0.2000000)

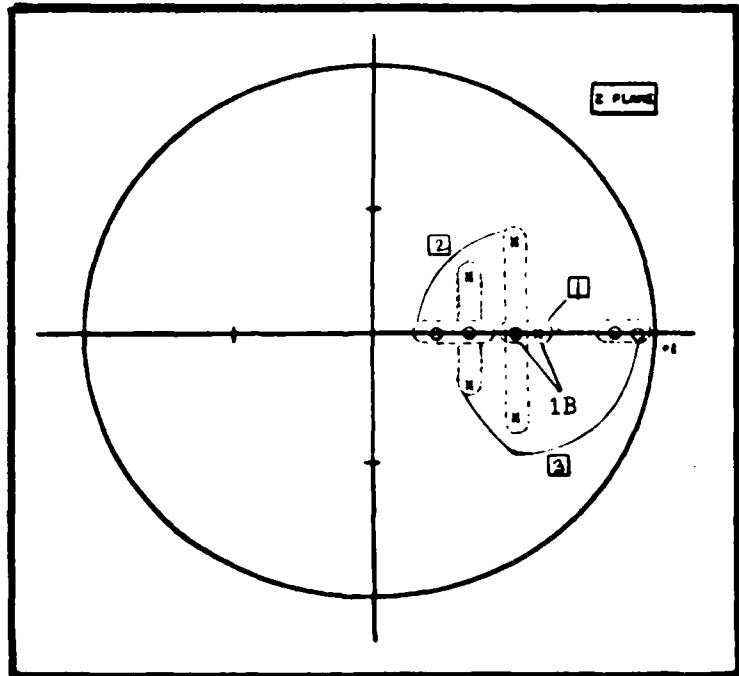
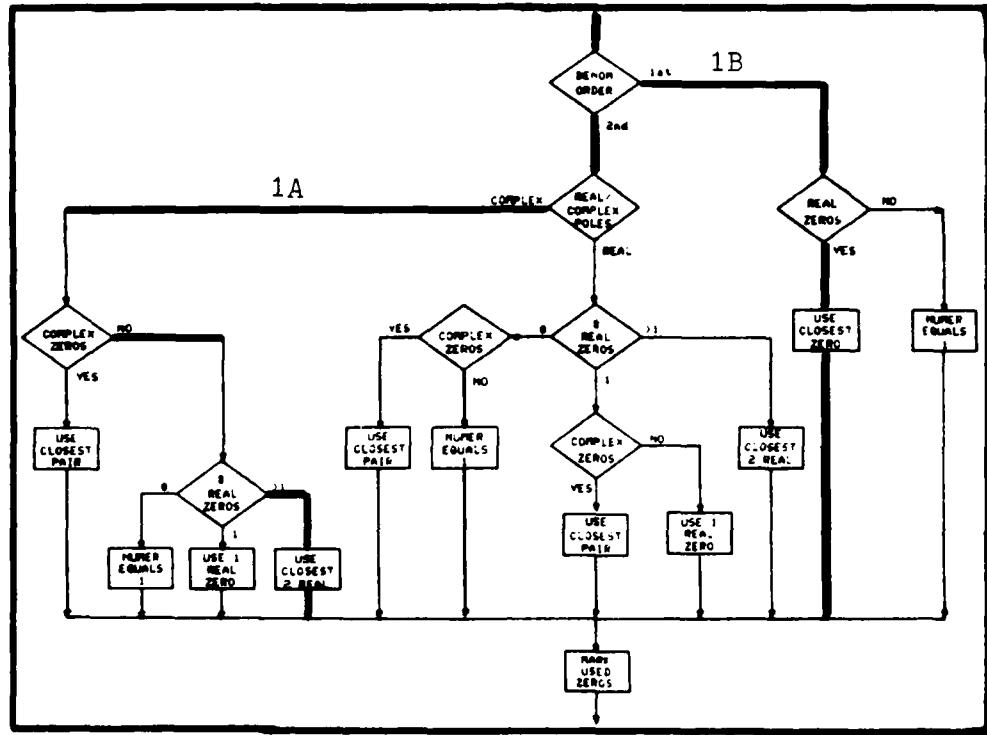


Figure D-2. Test Case 1A and 1B

TEST CASE 2A

OF SECTIONS 2

ZEROS

(0.3000000,0.1000000)
(0.3000000,-0.1000000)
(0.5000000,0.2000000)
(0.5000000,-0.2000000)

POLES

(0.5000000,0.3000000)
(0.5000000,-0.3000000)
(0.3000000,0.2000000)
(0.3000000,-0.2000000)

SECTION NUMBER = 1

ZEROS(1) = (0.5000000,0.2000000)
ZEROS(2) = (0.5000000,-0.2000000)
POLES(1) = (0.5000000,0.3000000)
POLES(2) = (0.5000000,-0.3000000)

SECTION NUMBER = 2

ZEROS(1) = (0.3000000,0.1000000)
ZEROS(2) = (0.3000000,-0.1000000)
POLES(1) = (0.3000000,0.2000000)
POLES(2) = (0.3000000,-0.2000000)

TEST CASE 2B

OF SECTIONS 3

ZEROS	POLES
(0.3000000,0.1000000)	(0.5000000,0.3000000)
(0.3000000,-0.1000000)	(0.5000000,-0.3000000)
(0.5000000,0.2000000)	(0.3000000,0.2000000)
(0.5000000,-0.2000000)	(0.3000000,-0.2000000)
(0.5000000,0.0000000E+00)	(0.6000000,0.0000000E+00)

SECTION NUMBER = 1

ZEROS(1) = (0.5000000,0.0000000E+00)
ZEROS(2) = (9999.000,9999.000)
POLES(1) = (0.6000000,0.0000000E+00)
POLES(2) = (9999.000,9999.000)

SECTION NUMBER = 2

ZEROS(1) = (0.5000000,0.2000000)
ZEROS(2) = (0.5000000,-0.2000000)
POLES(1) = (0.5000000,0.3000000)
POLES(2) = (0.5000000,-0.3000000)

SECTION NUMBER = 3

ZEROS(1) = (0.3000000,0.1000000)
ZEROS(2) = (0.3000000,-0.1000000)
POLES(1) = (0.3000000,0.2000000)
POLES(2) = (0.3000000,-0.2000000)

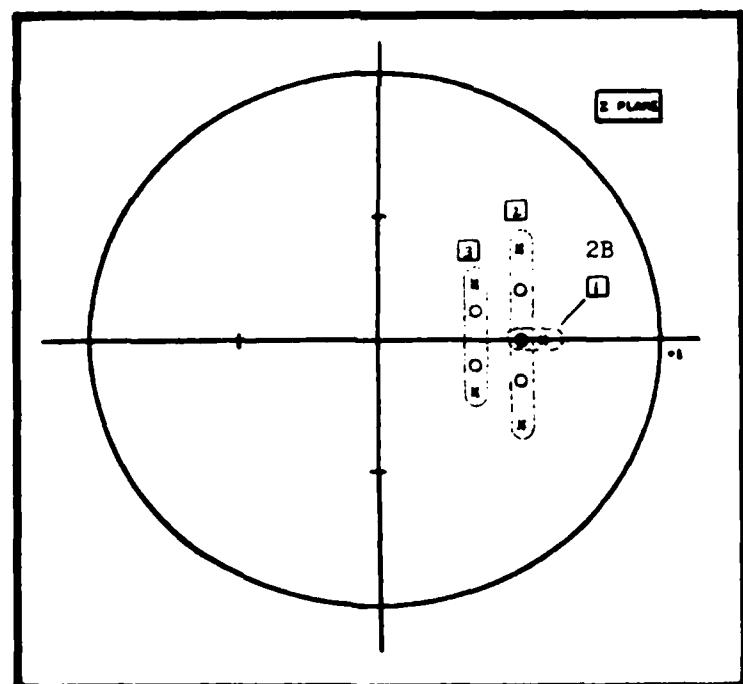
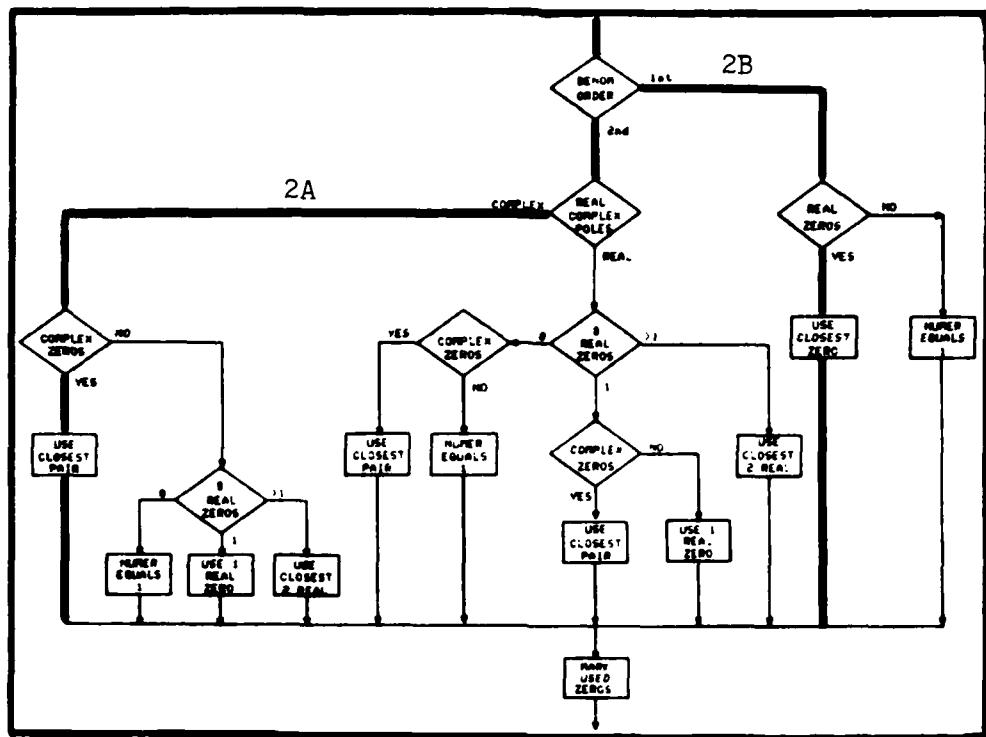


Figure D-3. Test Case 2A and 2B

TEST CASE 3A

OF SECTIONS 2

ZEROS

(0.3000000,0.1000000)
(0.3000000,-0.1000000)
(0.2000000,0.0000000E+00)
(0.3000000,0.0000000E+00)

POLES

(0.5000000,0.3000000)
(0.5000000,-0.3000000)
(0.3000000,0.2000000)
(0.3000000,-0.2000000)

SECTION NUMBER = 1

ZEROS(1) = (0.3000000,0.1000000)
ZEROS(2) = (0.3000000,-0.1000000)
POLES(1) = (0.5000000,0.3000000)
POLES(2) = (0.5000000,-0.3000000)

SECTION NUMBER = 2

ZEROS(1) = (0.3000000,0.0000000E+00)
ZEROS(2) = (0.2000000,0.0000000E+00)
POLES(1) = (0.3000000,0.2000000)
POLES(2) = (0.3000000,-0.2000000)

TEST CASE 3B

OF SECTIONS

3

ZEROS

(0.3000000,0.1000000)	POLES
(0.3000000,-0.1000000)	(0.5000000,0.3000000)
(0.2000000,0.0000000E+00)	(0.5000000,-0.3000000)
(0.3000000,0.0000000E+00)	(0.3000000,0.2000000)
(0.8000000,0.0000000E+00)	(0.3000000,-0.2000000)
	(0.6000000,0.0000000E+00)

SECTION NUMBER =

1

ZEROS(1) = (0.8000000,0.0000000E+00)
ZEROS(2) = (9999.000,9999.000)
POLES(1) = (0.6000000,0.0000000E+00)
POLES(2) = (9999.000,9999.000)

SECTION NUMBER =

2

ZEROS(1) = (0.3000000,0.1000000)
ZEROS(2) = (0.3000000,-0.1000000)
POLES(1) = (0.5000000,0.3000000)
POLES(2) = (0.5000000,-0.3000000)

SECTION NUMBER =

3

ZEROS(1) = (0.3000000,0.0000000E+00)
ZEROS(2) = (0.2000000,0.0000000E+00)
POLES(1) = (0.3000000,0.2000000)
POLES(2) = (0.3000000,-0.2000000)

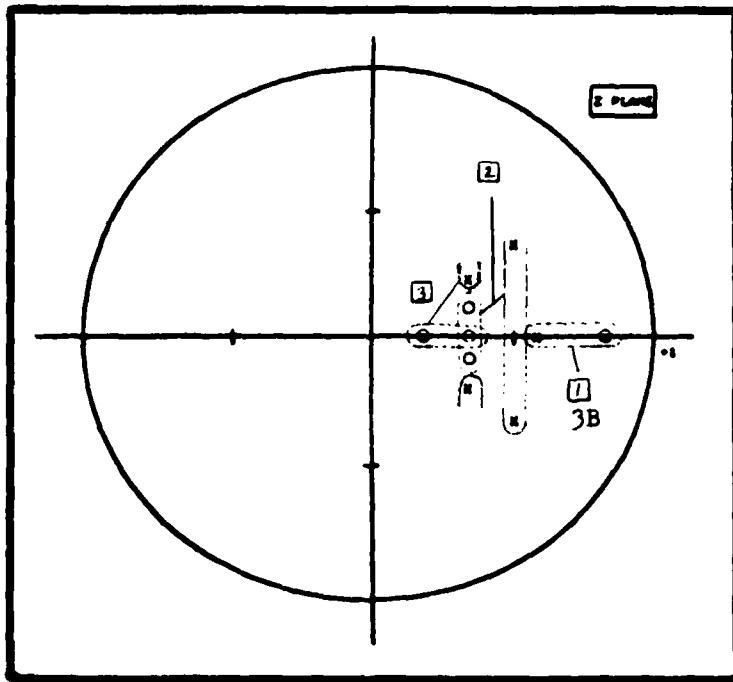
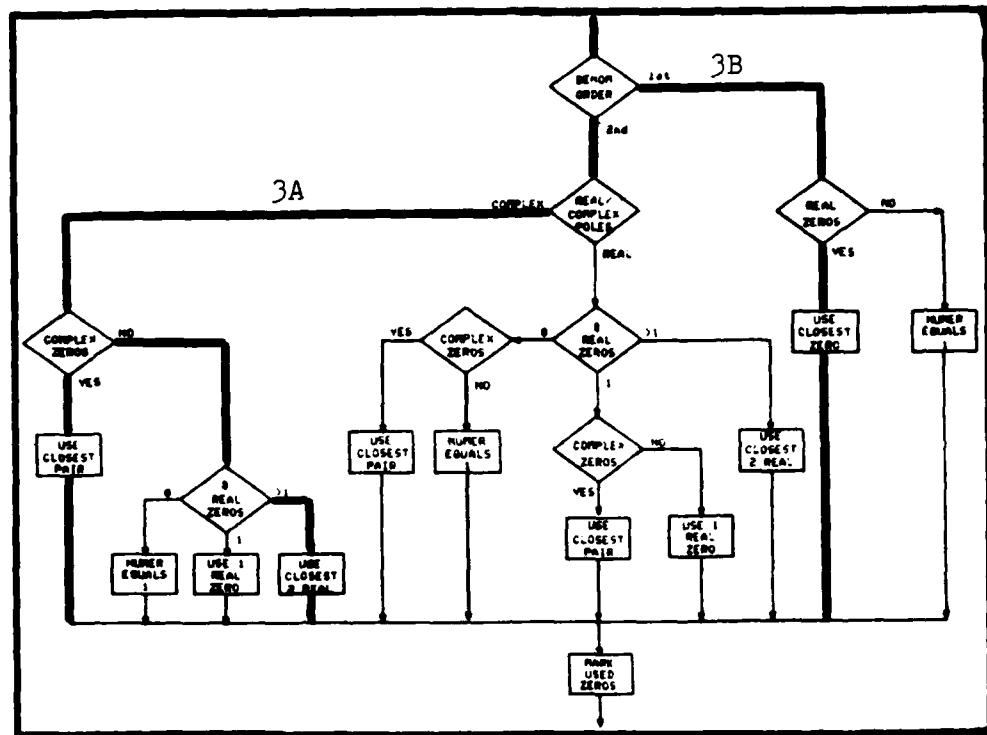


Figure D-4. Test Case 3A and 3B

TEST CASE 4A

OF SECTIONS **2**

ZEROS

(0.1000000,0.0000000E+00)
(0.3000000,0.0000000E+00)
(0.5000000,0.0000000E+00)
(0.7500000,0.0000000E+00)

POLES

(0.2000000,0.0000000E+00)
(0.3000000,0.0000000E+00)
(0.8000000,0.0000000E+00)
(0.9000000,0.0000000E+00)

SECTION NUMBER = **1**

ZEROS(1) = (0.7500000,0.0000000E+00)
ZEROS(2) = (0.1000000,0.0000000E+00)
POLES(1) = (0.9000000,0.0000000E+00)
POLES(2) = (0.2000000,0.0000000E+00)

SECTION NUMBER = **2**

ZEROS(1) = (0.5000000,0.0000000E+00)
ZEROS(2) = (0.3000000,0.0000000E+00)
POLES(1) = (0.8000000,0.0000000E+00)
POLES(2) = (0.3000000,0.0000000E+00)

TEST CASE 4B

OF SECTIONS 3

ZEROS	POLES
(0.1000000,0.0000000E+00)	(0.2000000,0.0000000E+00)
(0.3000000,0.0000000E+00)	(0.3000000,0.0000000E+00)
(0.5000000,0.0000000E+00)	(0.8000000,0.0000000E+00)
(0.7500000,0.0000000E+00)	(0.9000000,0.0000000E+00)
(0.7000000,0.0000000E+00)	(0.6000000,0.0000000E+00)

SECTION NUMBER = 1

ZEROS(1) = (0.7500000,0.0000000E+00)
ZEROS(2) = (0.1000000,0.0000000E+00)
POLES(1) = (0.9000000,0.0000000E+00)
POLES(2) = (0.2000000,0.0000000E+00)

SECTION NUMBER = 2

ZEROS(1) = (0.7000000,0.0000000E+00)
ZEROS(2) = (0.3000000,0.0000000E+00)
POLES(1) = (0.8000000,0.0000000E+00)
POLES(2) = (0.3000000,0.0000000E+00)

SECTION NUMBER = 3

ZEROS(1) = (0.5000000,0.0000000E+00)
ZEROS(2) = (9999.000,9999.000)
POLES(1) = (0.6000000,0.0000000E+00)
POLES(2) = (9999.000,9999.000)

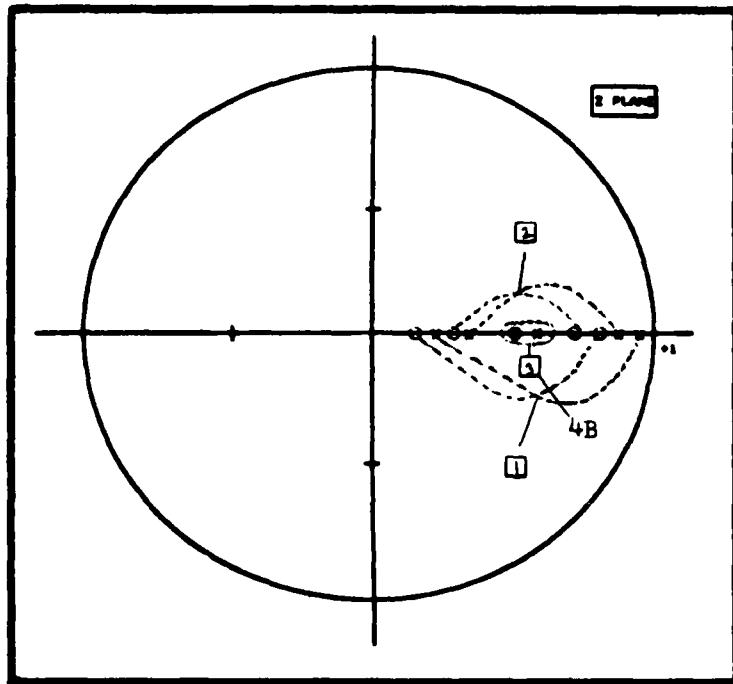
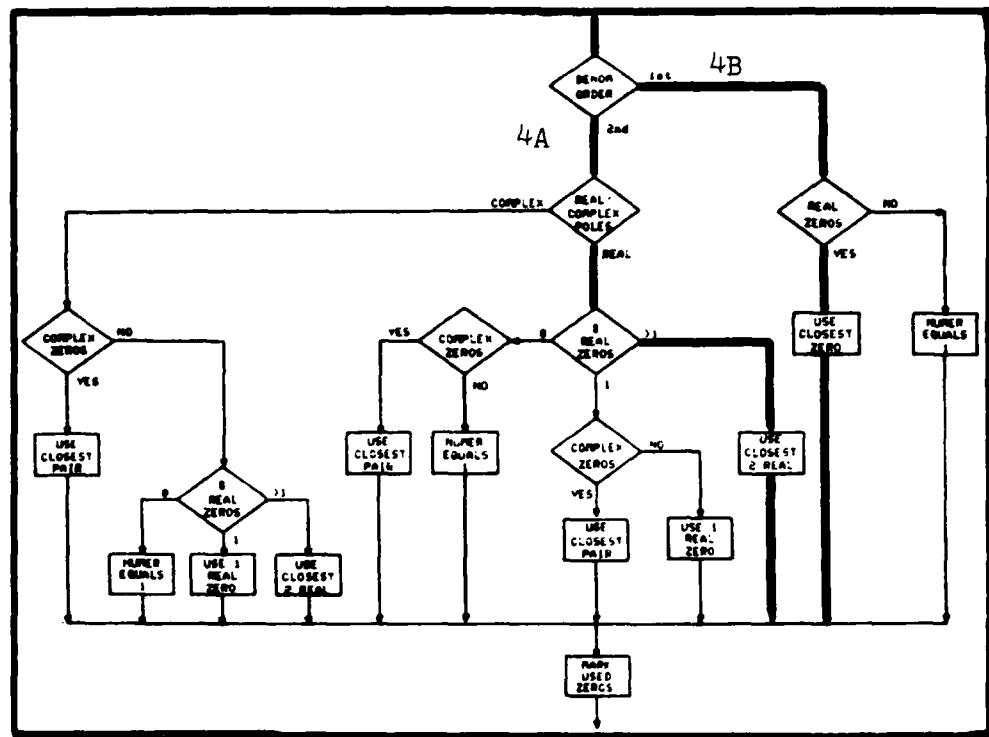


Figure D-5. Test Case 4A and 4B

TEST CASE 5A

OF SECTIONS 2

ZEROS

(0.3000000,0.2000000)
(0.3000000,-0.2000000)
(0.7000000,0.2000000)
(0.7000000,-0.2000000)

POLES

(0.2000000,0.0000000E+00)
(0.3000000,0.0000000E+00)
(0.8000000,0.0000000E+00)
(0.9000000,0.0000000E+00)

SECTION NUMBER = 1

ZEROS(1) = (0.7000000,0.2000000)
ZEROS(2) = (0.7000000,-0.2000000)
POLES(1) = (0.9000000,0.0000000E+00)
POLES(2) = (0.2000000,0.0000000E+00)

SECTION NUMBER = 2

ZEROS(1) = (0.3000000,0.2000000)
ZEROS(2) = (0.3000000,-0.2000000)
POLES(1) = (0.8000000,0.0000000E+00)
POLES(2) = (0.3000000,0.0000000E+00)

TEST CASE 5B

OF SECTIONS

3

ZEROS

(0.3000000,0.2000000)	POLES
(0.3000000,-0.2000000)	(0.2000000,0.0000000E+00)
(0.7000000,0.2000000)	(0.3000000,0.0000000E+00)
(0.7000000,-0.2000000)	(0.8000000,0.0000000E+00)
(0.7000000,0.0000000E+00)	(0.9000000,0.0000000E+00)

SECTION NUMBER =

1

ZEROS(1) =	(0.7000000,0.2000000)
ZEROS(2) =	(0.7000000,-0.2000000)
POLES(1) =	(0.9000000,0.0000000E+00)
POLES(2) =	(0.2000000,0.0000000E+00)

SECTION NUMBER =

2

ZEROS(1) =	(0.3000000,0.2000000)
ZEROS(2) =	(0.3000000,-0.2000000)
POLES(1) =	(0.8000000,0.0000000E+00)
POLES(2) =	(0.3000000,0.0000000E+00)

SECTION NUMBER =

3

ZEROS(1) =	(0.7000000,0.0000000E+00)
ZEROS(2) =	(9999.000,9999.000)
POLES(1) =	(0.6000000,0.0000000E+00)
POLES(2) =	(9999.000,9999.000)

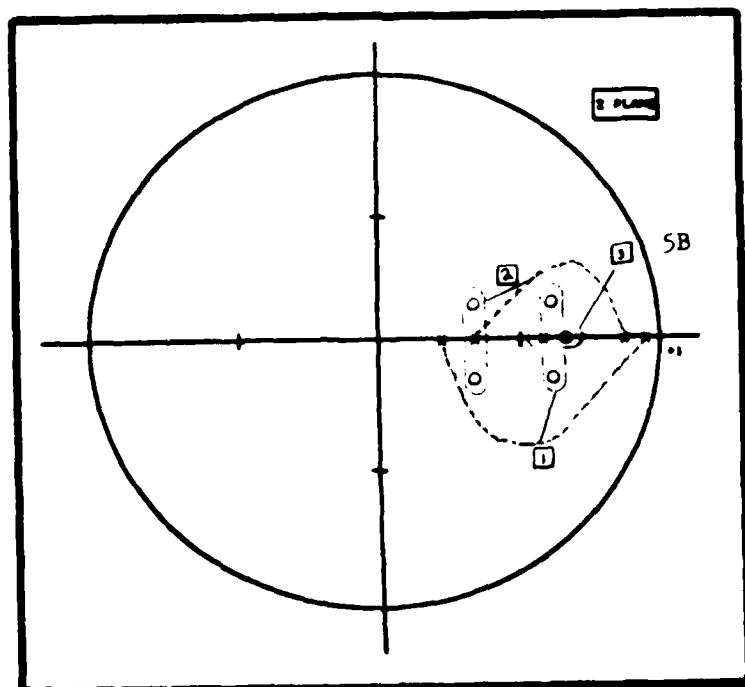
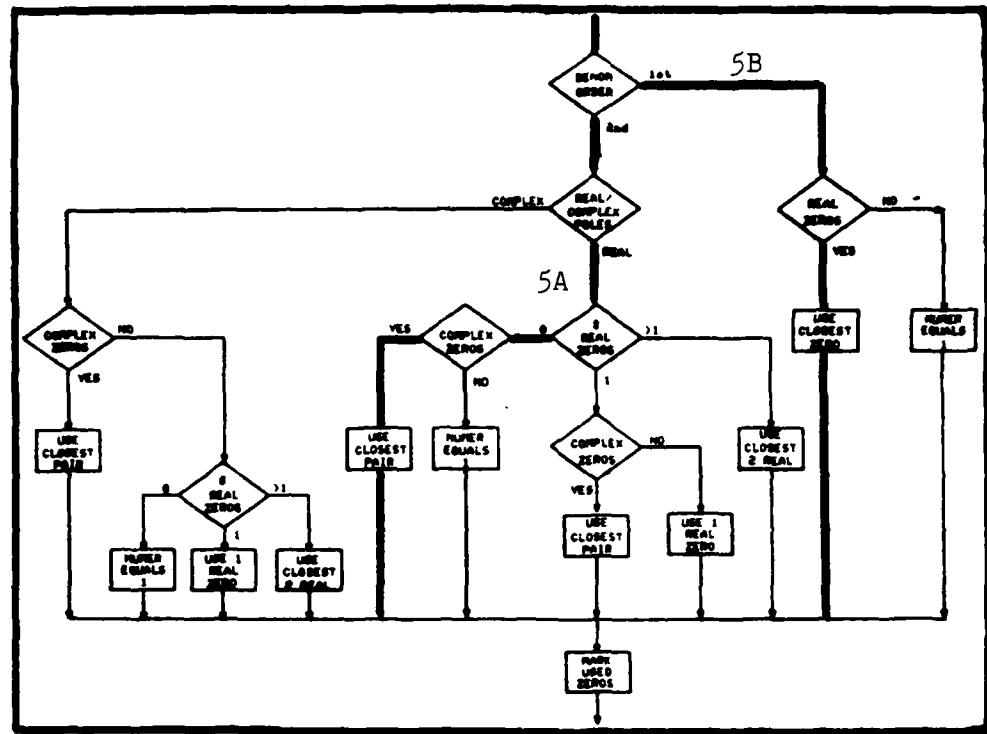


Figure D-6. Test Case 5A and 5B

TEST CASE 6A

OF SECTIONS 2

ZEROS

(0.3000000,0.2000000)
(0.3000000,-0.2000000)
(0.1000000,0.0000000E+00)
(0.3000000,0.0000000E+00)

POLES

(0.2000000,0.0000000E+00)
(0.3000000,0.0000000E+00)
(0.8000000,0.0000000E+00)
(0.9000000,0.0000000E+00)

SECTION NUMBER = 1

ZEROS(1) = (0.3000000,0.0000000E+00)
ZEROS(2) = (0.1000000,0.0000000E+00)
POLES(1) = (0.9000000,0.0000000E+00)
POLES(2) = (0.2000000,0.0000000E+00)

SECTION NUMBER = 2

ZEROS(1) = (0.3000000,0.2000000)
ZEROS(2) = (0.3000000,-0.2000000)
POLES(1) = (0.8000000,0.0000000E+00)
POLES(2) = (0.3000000,0.0000000E+00)

TEST CASE 6B

OF SECTIONS 3

ZEROS

(0.3000000,0.2000000)	POLES
(0.3000000,-0.2000000)	(0.2000000,0.0000000E+00)
(0.1000000,0.0000000E+00)	(0.3000000,0.0000000E+00)
(0.3000000,0.0000000E+00)	(0.8000000,0.0000000E+00)
(0.5000000,0.0000000E+00)	(0.9000000,0.0000000E+00)

SECTION NUMBER = 1

ZEROS(1) =	(0.5000000,0.0000000E+00)
ZEROS(2) =	(0.1000000,0.0000000E+00)
POLES(1) =	(0.9000000,0.0000000E+00)
POLES(2) =	(0.2000000,0.0000000E+00)

SECTION NUMBER = 2

ZEROS(1) =	(0.3000000,0.2000000)
ZEROS(2) =	(0.3000000,-0.2000000)
POLES(1) =	(0.8000000,0.0000000E+00)
POLES(2) =	(0.3000000,0.0000000E+00)

SECTION NUMBER = 3

ZEROS(1) =	(0.3000000,0.0000000E+00)
ZEROS(2) =	(9999.000,9999.000)
POLES(1) =	(0.6000000,0.0000000E+00)
POLES(2) =	(9999.000,9999.000)

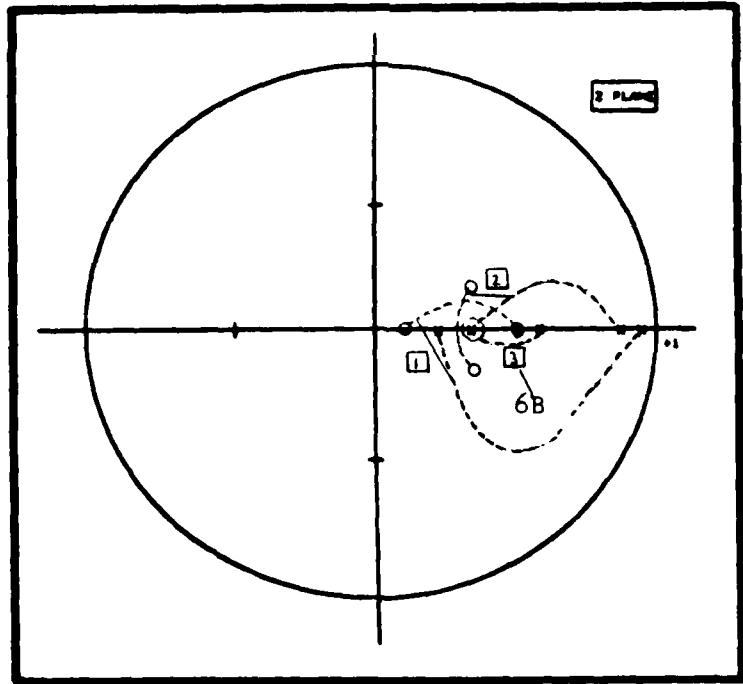
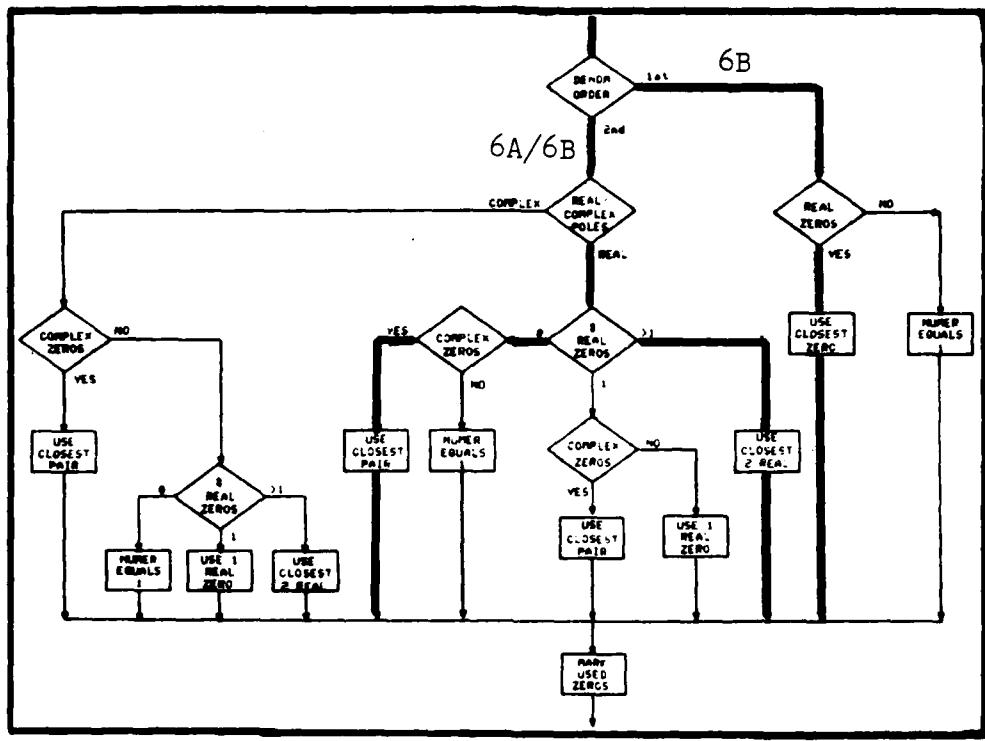


Figure D-7. Test Case 6A and 6B

Module: RDHTF.THS

This module reads the controller parameters into the main program from the data file MEMORY.DAT. The MEMORY.DAT file is a sequential data file that must be read until the HTF variable (variable from ICECAP where controller parameters are stored) is encountered. The variable TSAMP, which contains the sample period, is also read into DICES.

A design session using ICECAP is executed in order to store a controller transfer function and sample-period in the file MEMORY.DAT. This file can be examined using a text editor.

The transfer function used for this test is shown below (TSAMP = .1234):

FEEDBACK-LOOP TRANSFER FUNCTION (HTF)

$$HK = (HNK / HDK) = 0.1250$$

HTF(S) NUMERATOR

I	HNPOLY(I)	HZERO(I)
1	(0.1250)S** 4	(0.5430) + J(0.0000E+00)
2	(-0.2468E-01)S** 3	(0.1230) + J(0.0000E+00)
3	(-0.1712E-01)S** 2	(-0.2343) + J(0.2311)
4	(-0.5104E-02)S** 1	(-0.2343) + J(-0.2311)
5	(0.9042E-03)	HNK = 0.1250

HTF(S) DENOMINATOR

I	HDPOLY(I)	HPOLE(I)
1	(1.000)S** 5	(0.2323) + J(0.0000E+00)
2	(-1.121)S** 4	(0.1232) + J(0.2123)
3	(0.5732)S** 3	(0.1232) + J(-0.2123)
4	(-0.1604)S** 2	(0.3212) + J(0.2122)
5	(0.2640E-01)S** 1	(0.3212) + J(-0.2122)
6	(-0.2074E-02)	HDK = 1.000

Table D-1. Test Transfer Function

The portion of MEMORY.DAT to be read follows:

350	.0000000	.0000000	.5430000	.1230000	-.2343000
351	-.2343000	.0000000	.0000000	.0000000	.0000000
352	.0000000	.0000000	.0000000	.0000000	.0000000
353	.0000000	.0000000	.0000000	.0000000	.0000000
354	.0000000	.0000000	.0000000	.0000000	.0000000
355	.0000000	.0000000	.0000000	.0000000	.0000000
356	.0000000	.0000000	.0000000	.0000000	.0000000
357	.0000000	.0000000	.0000000	.0000000	.0000000
358	.0000000	.0000000	.0000000	.0000000	.0000000
359	.0000000	.0000000	.0000000	.0000000	.0000000
360	.0000000	.0000000	.0000000	.0000000	.2311000
361	-.2311000	.0000000	.0000000	.0000000	.0000000
362	.0000000	.0000000	.0000000	.0000000	.0000000
363	.0000000	.0000000	.0000000	.0000000	.0000000
364	.0000000	.0000000	.0000000	.0000000	.0000000
365	.0000000	.0000000	.0000000	.0000000	.0000000
366	.0000000	.0000000	.0000000	.0000000	.0000000
367	.0000000	.0000000	.0000000	.0000000	.0000000
368	.0000000	.0000000	.0000000	.0000000	.0000000
369	.0000000	.0000000	.0000000	.0000000	.0000000
370	.0000000	.0000000	.2323000	.1232000	.1232000
371	.3212000	.3212000	.0000000	.0000000	.0000000
372	.0000000	.0000000	.0000000	.0000000	.0000000
373	.0000000	.0000000	.0000000	.0000000	.0000000
374	.0000000	.0000000	.0000000	.0000000	.0000000
375	.0000000	.0000000	.0000000	.0000000	.0000000
376	.0000000	.0000000	.0000000	.0000000	.0000000
377	.0000000	.0000000	.0000000	.0000000	.0000000
378	.0000000	.0000000	.0000000	.0000000	.0000000
379	.0000000	.0000000	.0000000	.0000000	.0000000
380	.0000000	.0000000	.0000000	.2123000	-.2123000
381	.2122200	-.2122200	.0000000	.0000000	.0000000
382	.0000000	.0000000	.0000000	.0000000	.0000000
383	.0000000	.0000000	.0000000	.0000000	.0000000
384	.0000000	.0000000	.0000000	.0000000	.0000000
385	.0000000	.0000000	.0000000	.0000000	.0000000
386	.0000000	.0000000	.0000000	.0000000	.0000000
387	.0000000	.0000000	.0000000	.0000000	.0000000
388	.0000000	.0000000	.0000000	.0000000	.0000000
389	.0000000	.0000000	.0000000	.0000000	.0000000
390	.0000000	.0000000	4	5	.1250000
391	.1250000	1.000000	1.000000		
392	1.000000	.0000000	.0000000	.0000000	.0000000
393	.0000000	.0000000	.0000000	.0000000	.0000000

Table D-2. Partial MEMORY.DAT Data File

```
613 .0000000 .0000000 .0000000 .0000000  
614 1.000000 .0000000 0 0 0 0  
615 9.999998E-03 .0000000 0 .1234000
```

```
:  
:  
:  
:
```

Table D-2 (cont). Partial MEMORY.DAT Data File

The module is executed and the variable within RDHTF.THS which contains the controller parameters is displayed.

CZERO

```
(0.5430000,0.0000000E+00)  
(0.1230000,0.0000000E+00)  
(-0.2343000,0.2311000)  
(-0.2343000,-0.2311000)
```

CPOLE

```
(0.2323000,0.0000000E+00)  
(0.1232000,0.2123000)  
(0.1232000,-0.2123000)  
(0.3212000,0.2122200)  
(0.3212000,-0.2122200)
```

TSAMP

.123400

Table D-3. Data Read From MEMORY.DAT

It can be seen that the coefficients and sample-period are properly read.

Module: MULTRTS.THS

This module multiplies the roots and gains to produce the second-order polynomials that will be implemented in the digital controller. Each coefficient is then quantized and converted to a 16-bit integer to be used within the TMS32010 microprocessor.

The following table shows the second-order sections and the quantized coefficients along with the 16-bit integer equivalent.

# OF SECTIONS	3
NUMERATOR ORDER	4
DENOMINATOR ORDER	5
CZERO	
(0.5430000,0.0000000E+00)	
(0.1230000,0.0000000E+00)	
(-0.2343000,0.2311000)	
(-0.2343000,-0.2311000)	
CPOLE	
(0.2323000,0.0000000E+00)	
(0.1232000,0.2123000)	
(0.1232000,-0.2123000)	
(0.3212000,0.2122200)	
(0.3212000,-0.2122200)	

SECTION NUMBER =	1

CORDPZ ZEROS(1) = (-0.2343000,0.2311000)	
CORDPZ ZEROS(2) = (-0.2343000,-0.2311000)	
CORDPZ POLES(1) = (0.3212000,0.2122200)	
CORDPZ POLES(2) = (0.3212000,-0.2122200)	

SECTION NUMBER =	2

CORDPZ ZEROS(1) = (0.1230000,0.0000000E+00)	
CORDPZ ZEROS(2) = (9999.000,9999.000)	
CORDPZ POLES(1) = (0.2323000,0.0000000E+00)	
CORDPZ POLES(2) = (9999.000,9999.000)	

SECTION NUMBER =	3

CORDPZ ZEROS(1) = (0.5430000,0.0000000E+00)	
CORDPZ ZEROS(2) = (9999.000,9999.000)	
CORDPZ POLES(1) = (0.1232000,0.2123000)	
CORDPZ POLES(2) = (0.1232000,-0.2123000)	

Table D-4. Sections of Filter

QUANTIZATION ROUTINES

NUMSECTS = 3
ADJ = 1.000010132789612
HK = 0.125000036663376
FRACHK = 0.499999945597523
TSAMP = 0.1234000

SECTION NUMBER 1

B0 = 0.5000050609545031
B1 = 0.2343023740190164
B2 = 5.4152399803970321E-02
IB0 = 16384
IB1 = 7678
IB2 = 1774
A1 = 0.6424000263214111
A2 = -0.1482067853212357
IA1 = 21050
IA2 = -4855

SECTION NUMBER 2

B0 = 0.5000050609545031
B1 = -6.1500624196153455E-02
IB0 = 16384
IB1 = -2014
A1 = 0.2322999984025955
IA1 = 7612

SECTION NUMBER 3

B0 = 0.5000050609545031
B1 = -0.2715027395151395
IB0 = 16384
IB1 = -8896
A1 = 0.2463999986648560
A2 = -6.0249533504247665E-02
IA1 = 8074
IA2 = -1973

TSAMP = 0.1234000
ITSAMP = 170

Table D-5. Quantized Filter Parameters

Module: NEWCO.THS

This module inserts the new coefficients and sample-period constant into the source code for the controller program (3DFILT.THS).

The module MULTRTS.THS form a data file called NEWCO.THS which contain the new filter coefficients and the sample period constant. The file NEWCO.THS must be checked prior to this test to ensure its correct operation. This check can be made using the VMS text editor to inspect the data file for the proper contents.

The file NEWCO.THS contains the new coefficients:

16384
-2014
0
7612
0
16384
-8896
0
8074
-1973
16384
7678
1774
21050
-4855
32767
0
0
0
0
170

Table D-6. File of New Data

The 3DFILT.THS section to be modified is shown below.

CB01	DATA	32767
CB11	DATA	0
CB21	DATA	0
CA11	DATA	0
CA21	DATA	0
CB02	DATA	32767
CB12	DATA	0
CB22	DATA	0
CA12	DATA	0
CA22	DATA	0
CB03	DATA	32767
CB13	DATA	0
CB23	DATA	0
CA13	DATA	0
CA23	DATA	0
CB04	DATA	32767
CB14	DATA	0
CB24	DATA	0
CA14	DATA	0
CA24	DATA	0
SMP	DATA	800

Table D-7. Original Filter Data

After execution of the module CHGCO.THS the new data
is now properly located in the source file.

CB01	DATA	16384
CB11	DATA	-2014
CB21	DATA	0
CA11	DATA	7612
CA21	DATA	0
CB02	DATA	16384
CB12	DATA	-8896
CB22	DATA	0
CA12	DATA	8074
CA22	DATA	-1973
CB03	DATA	16384
CB13	DATA	7678
CB23	DATA	1774
CA13	DATA	21050
CA23	DATA	-4855
CB04	DATA	32767
CB14	DATA	0
CB24	DATA	0
CA14	DATA	0
CA24	DATA	0
SMP	DATA	170

Table D-8. Modified Filter Data

Test Summary: All coefficients and the sample-period constant were properly read in and placed in the source file.

Appendix E - Program Listings

Program Listings

This appendix contains the main DICES VAX FORTRAN program listing and the TMS320 Assembler List File for the generic 3D filter program. The DICES listing is E.1 and the TMS32010 listing is E.2.

Appendix E.1 - DICES Program Listing

```

C*****
C
C          DIGITAL INTERACTIVE CONTROLLER
C
C          EVALUATION SYSTEM (DICES)
C
C          CAPT SCOTT ECKERT
C
C***** PROGRAM DICES
C***** C PROGRAM NAME: DICES.THS
C
C AUTHOR: CAPT. SCOTT B. ECKERT/GE-85D
C
C DATE: 4 NOVEMBER, 1985
C
C FUNCTION: DICES PERMITS THE IMPLEMENTATION OF A DIGITAL
C CONTROLLER DESIGNED USING THE ICECAP CAD PKG. IT THEN
C PERMITS THE EXECUTION OF THE FILTER ON A TMS32010
C MICROPROCESSOR. THE CONTROLLER IS THEN INSERTED INTO A
C CLOSED-LOOP SIMULATION USING AN ANALOG COMPUTER AND
C PERFORMANCE TESTING IS THEN PERFORMED. DICES CONFIGURES AND
C RUNS THE TESTS ON THE CLOSED-LOOP SYSTEM.
C*****
COMPLEX CPOLE(8),CZERO(8),CORDPZ(16)
INTEGER ISECTOR(4),STIME
CHARACTER=5 FROSPAN,FRC,DUR
CHARACTER RESP=3, FORM=6, FORM1=6, DNAM=3           !FMS
CHARACTER FILE=30, DATA=255                         !FMS
CHARACTER=1 OPP                                     !FMS
CHARACTER=3 ANPC,ANPCC,OPP3
CHARACTER=2 OPP2
CHARACTER=6 OPP6

C THIS SECTION SETS UP USE OF FMS
C
DIMENSION IMPURE (32)
CALL FDV$INIT (*DESCR(IMPURE), *REF(1000))
CHAN=2
CALL FDV$LCHAN(CHAN)
CALL FDV$OPEN('DLIB',1)                            !OPEN FORM LIBRARY

C FMS FORMS USED
C MENU1      OPENING MAIN MENU
C MENU2      PERFORMANCE SPECS
C MENU3      MATH MODEL DEVELOPMENT
C MENU4      IMPLEMENT CONTROLLER MENU
C MENU6      ENTERING ICECAP MESSAGE
C MENU7      'DICES' BANNER
C MENU8      STEP TEST AMP AND SETTLING TIME REQUEST
C MENU9/MENU10
C MENU16     CTRL T CTRL START MESSAGE TO BEGIN TEST
C MENU17     STEP TEST IN PROGRESS

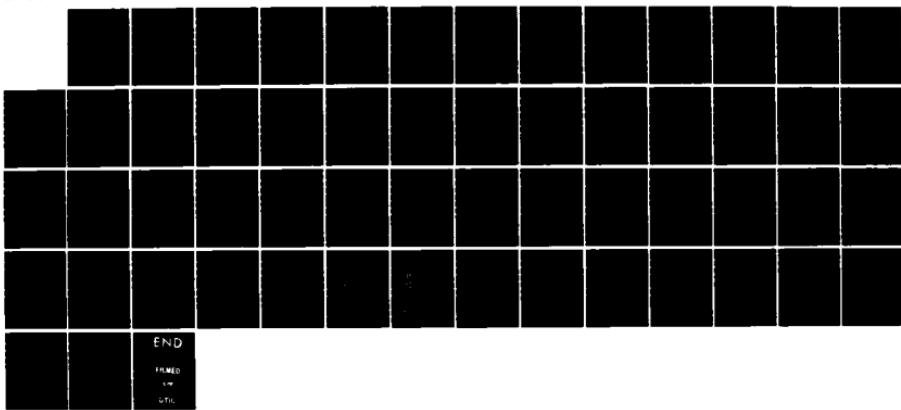
```

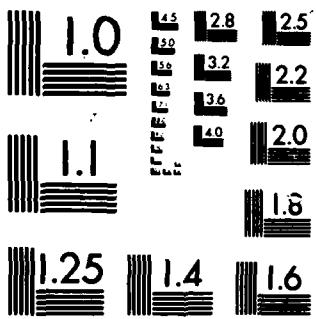
AD-A163 966 DEVELOPMENT OF A DIGITAL INTERACTIVE CONTROLLER
EVALUATION SYSTEM (DICES)(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
UNCLASSIFIED S B ECKERT DEC 85 AFIT/GE/ENG/85D-13

5/5

F/G 9/5

NL





MICROCOPY RESOLUTION TEST CHART

U.S. NATIONAL BUREAU OF STANDARDS 1963-A

```

C MENU18      TMS32010 ASSEMBLER SOURCE FILE MESSAGE
C MENU19      REPORT PRINTING MENU
C MENU20      RETURNING TO VMS MESSAGE
C MENU21      FREQ RESPONSE TEST IN PROGRESS
C*****MENU*****C
C CALL SUB TO SEE IF EQUIPMENT IS TO BE INITIALIZED
C IF B&K AND 172B ARE NOT TURNED ON- MUST SKIP
C INITIALIZATION SUBROUTINE
C
C DISPLAY DICES MESSAGE WHILE EQUIPMENT IS BEING CONFIGURED
    CALL FDV$CDISP('MENU7')          !FMS 'MENU7' IS 'DICES'
    J=IBUPU(0,2,0)                  !CLR ALL INSTR ON IEEE BUS
    J=IBUPU(0,4,2)
    CALL CONFIGEQUIP                !INITIALIZE EQUIPMENT ON BUS
    GENFLG=0                         !SET GENERATOR FLAG TO 0
C*****MENU*****C
10     CALL FDV$CDISP('MENU1')
        CALL FDV$GETAL(0PP,TERMINATOR,'CHOICE')
        CALL CHARTONUM(0PP,IOPT1)      !CONVERT RETURNED CHAR TO NUMBER
C
C OPTIONS ARE:
C 1. SYSTEM PERFORMANCE SPECS
C 2. MATH MODEL DEVELOPMENT
C 3. PLANT SIMULATION
C 4. CONTROLLER DESIGN
C 5. IMPLEMENT CONTROLLER
C 6. PERFORMANCE TESTING
C 7. PRINT/DELETE PAIRING LIST
C 8. EXIT PROGRAM
20     IF ((IOPT1 .LT. 1) .OR. (IOPT1 .GT. 8)) THEN
          GO TO 10
        ENDIF
C
C GO TO CORRECT SUB-MENU
GOTO(100,200,300,400,500,600,700,9999)IOPT1
C*****MENU*****C
C SYSTEM PERFORMANCE SPECS MESSAGE USING FMS
100    CALL FDV$CDISP('MENU2')
        CALL FDV$GETAL(0PP,TERMINATOR,'CHOICE')
        GO TO 10
C*****MENU*****C
C MATH MODEL DEVELOPMENT MESSAGE
200    CALL FDV$CDISP('MENU3')
        CALL FDV$GETAL(0PP,TERMINATOR,'CHOICE')
        GO TO 10
C*****MENU*****C
C PLANT SIMULATION
300    GO TO 10
C*****MENU*****C
C CONTROLLER DESIGN
C 'MENU6' DISPLAYS MESSAGE THAT PROGRAM IS ENTERING ICECAP.
C
400    CALL FDV$CDISP('MENU6')          !FMS 'MENU6'
C        CALL FDV$GETAL(0PP,TERMINATOR,'CHOICE')
        IST=LIB$SPAWN ('@{ICECAP1 MODULES}RICEP')

```

```

C
400    CALL FDV$CDISP('MENU5')           !FMG 'MENU5'
C     CALL FDV$BETAL(OPP, TERMINATOR, 'CHOICE')
C     IST=LIB$SPAWN ('@[ICECAP1 MODULES]RICER')
C     GO TO 10
C*****+
C IMPLEMENT CONTROLLER
C
C     SUB-MENU
500    CALL FDV$CDISP('MENU4')
     CALL FDV$BETAL(OPP, TERMINATOR, 'CHOICE')
     CALL CHARTONUM(OPP, IOPT2)      !CONV RETURNED CHAR TO NUMBER
C*****+
IF ((IOPT2 .LT. 1) .OR. (IOPT2 .GT. 6)) THEN
    GO TO 500
ENDIF
BOTO(510, 520, 530, 540, 550, 560) IOPT2
C*****+
C READ DESIGN PARAMETERS FROM DATA FILE MEMORY.DAT
C OPTION 5-1
510    CALL CLEARSCREEN
PRINT#
PRINT#, 'READING ORDER, BAIN, AND TSAMP'
CALL RDORDER(INORD, IDORD, HK, TSAMP)
IF (TSAMP .EQ. 0) THEN
    CALL CLEARSCREEN
    PRINT#, '      ERROR !!!'
    PRINT#
    PRINT#, '**** TSAMP = 0 ****'
    PRINT#
    PRINT#
512    PRINT#, 'YOU MUST ENTER A VALUE OTHER THAN ZERO FOR TSAMP'
    PRINT#, 'ENTER TSAMP:'
    READ#, TSAMP
    IF (TSAMP .EQ. 0) THEN
        GO TO 512
    ENDIF
    PRINT#, 'CONTROLLER ORDER READ'
ENDIF
PRINT#, 'READING DESIGN PARAMETERS'
CALL RDHTF(INORD, IDORD, CZERO, CPOLE)
PRINT#, 'PAIRING POLES AND ZEROS'
CALL PAIR(HK, INORD, IDORD, CZERO, CPOLE, CORDPZ,
+ NUMSECTB, ISECTORD)
GO TO 500
C*****+
C ROUND-OFF OR TRUNCATE
C*****+
520    CALL FDV$CDISP('MENU13')          !FMG MENU 13
     CALL FDV$BETAL(OPP, TERMINATOR, 'CH1')
     CALL CHARTONUM(OPP, TRFLB)      !CONV RETURNED CHAR TO NUMBER
IF (TRFLB .EQ. 1) THEN
    TRFLB=0                      !ROUND-OFF
ELSEIF (TRFLB .EQ. 2) THEN
    TRFLB=1                      !TRUNCATE

```

```

        ENDIF
        GO TO 500
C*****+
530    PRINT*
C
C CHECK TO SEE IF OPTION 1 WAS NOT EXECUTED FIRST.
C THIS IS DONE BY CHECKING THE VARIABLE HK.
C IF HK=0, THEN IT WAS NOT READ IN BY RDOORDER.THS
        IF (HK .EQ. 0) THEN
            CALL FDV$CDISP('MENU12')
            CALL FDV$BETAL(OPP, TERMINATOR, 'CH1')
            GO TO 500
        ENDIF
        CALL CLEARSCREEN
        PRINT*, 'MULTIPLYING ROOTS AND QUANTIZING COEFFICIENTS'
        CALL MULTRTS(CORDPZ, NUMSECTS, TSAMP, HK, ISECTORD, TRFLB)
        PRINT*, 'CHANGING COEFFICIENTS IN TMB32010 SOURCE CODE'
        CALL CHGCO
        GO TO 500
C*****+
C CALL TMB32010 ASSEMBLER
540    CALL FDV$CDISP('MENU18')
        IST=LIB$SPAWN ('X320')
        GO TO 500
C*****+
C LOAD TMB32010 WITH OBJECT CODE
C
550    CALL FDV$CDISP('MENU20')           !RETURN TO VMS MESSAGE
        GO TO 10000
C*****+
560    GO TO 10           !RETURN TO MAIN MENU
C*****+
C PERFORMANCE TESTING MENU
C
C     SUB-MENU
580    CALL FDV$CDISP('MENU5')           !USE 'MENU5' FOR MENU
        CALL FDV$BETAL(OPP, TERMINATOR, 'CHOICE')
        CALL CHARTONUM(OPP, IOPT7)      !CONVERT RETURNED CHAR TO NUMBER
C
C     CHECK OPTION NUMBER
C
        IF ((IOPT7 .LT. 1) .OR. (IOPT7 .GT. 4)) THEN
            GO TO 600
        ENDIF
        GOTO(610, 620, 630, 640) IOPT7
C*****+
C OPTION 1 - STEP RESPONSE
C*****+
C*****+
610    CALL WRITEMSG(2, 'EN 68,0', ISTAT)      !DISABLE GENERATOR
C*****+
C DISPLAY MESSAGE - ONE MOMENT PLEASE
C
        CALL FDV$CDISP('MENU15')

```

```

C*****
C
C
    CALL STEPSET           !SET UP B&K FOR STEP TEST
    CALL WRITEMSG(2,'KP H',IBSTAT)   !SINGLE RECORD MODE
612   CALL FDV$CDISP('MENU8')
    CALL FDV$SET(OPP2,TERMINATOR,'CH1')   !OPP2 = 2 CHAR
    CALL TWOCHAR(OPP2,AMP)           !TWO CHARS TO NUNS
C CHECK LIMITS
    IF ((AMP .LT. 0) .OR. (AMP .GT. 7.5)) THEN
        GOTO 612
    ENDIF
C
    CALL CONVCHAR(AMP,AMPC) !CONVERT TO (X,X) CHARACTER
    CALL SETBKINLEVEL(AMP)   !SET B&K MAX INPUT LEVEL
    CALL FDV$SET(OPP3,TERMINATOR,'CH2')   !OPP3 = 3 CHAR
    CALL THREECHAR(OPP3,STIME)          !CONV 3 CHARS TO INT
C
    !STIME IS AN INT
C*****
C
    IF ((STIME .LT. .015) .OR. (STIME .GT. .512)) THEN
        CALL CLEARSCEEN
        PRINT*, 'YOU HAVE EXCEEDED THE LIMITS OF DICES !!!'
        PRINT*
        PRINT*, 'THE SETTLING TIME MUST BE BETWEEN .015 SEC'S'
        PRINT*, 'AND .512 SEC'S.'
        PRINT*
        PRINT*
        CALL WAIT
        GO TO 612
    ELSE
        CALL CALCFREQSPAN(STIME,FREQSPAN,ONTIME)
    ENDIF
C
C*****
C B&K 2032 SET UP
C WRITE FREQ SPAN TO B&K
    CALL WRITEMSG(2,'EM FS',//FREQSPAN,IBSTAT)
C
C WRITE Y AMPLITUDE TO B&K
    CALL CONVCHAR(AMP+.5,AMPC)      !ADD .5 VOLTS
    CALL WRITEMSG(2,'ED YF',//AMPC,IBSTAT)  ! WRITE TO B&K
C
    CALL WRITEMSG(2,'DF UF',IBSTAT)  !FULL DISPLAY
C
C DISPLAY MESSAGE TO USER TO PREPARE TO START TEST
    CALL FDV$CDISP('MENU16')       !CTRL T MSG/CTRL START
    CALL WRITEMSG(2,'CP IF',IBSTAT) !INV FLASH VIDEO B&K
    CALL WRITEMSG(2,'PR "PRESS #CONTROL#START# TO BEGIN TEST...
+
    ",IBSTAT)
    CALL WAITSTART
    CALL FDV$CDISP('MENU17')       !STEP TEST IN PROGRESS
    CALL WRITEMSG(2,'PR "
+
    ",IBSTAT)
    CALL WRITEMSG(2,'PR "MEASUREMENT IN PROGRESS....'
+
    ",IBSTAT)

```

```

C*****
C START TEST
C*****
C WAVETEK 172B SET UP
C*****
    CALL CONVCHAR(AMP,AMPCC)           !CONV TO CHAR DATA
    CALL WRITEMSG(3,'D0B0C4P0I\n',ISTAT) !S V DC MODE
    CALL WRITEMSG(3,'D//AMPCC//I',ISTAT) !TURN 172B ON
C*****
    CALL WAITPERIOD(ONTIME)          !WAIT FOR TIME TO PASS
    CALL WAITPERIOD(1.0)              !WAIT EXTRA 2 SECONDS
C                                     !ONTIME COMES FROM CALCFCRQSPAN SUB
C DONE WITH TEST
    CALL WRITEMSG(3,'DBI',ISTAT)      !TURN OFF 172B OUTPUT
    J=IBUPU(0,2,0)                   !CLEAR B&K 2632
    J=IBUPU(0,5,2)                   !B&K TO LOCAL
C
    CALL WRITEMSG(2,'SR 1',ISTAT)     !AUTO SCALE Y AXIS
    CALL WRITEMSG(2,'KP D',ISTAT)     !MOVE TO CURSOR FIELD
    CALL WRITEMSG(2,'KP S',ISTAT)     !MOVE CURSOR DOWN
    CALL WRITEMSG(2,'KP -',ISTAT)     !MOVE CURSOR DOWN
C RETURN TO MENU
    CALL WRITEMSG(2,'WT 9,1,"'
    +
    " ",ISTAT)
    CALL WRITEMSG(2,'SR',ISTAT)
    CALL WRITEMSG(2,'PR "TEST COMPLETE.",',ISTAT)
    J=IBUPU(0,5,2)                   !B&K TO LOCAL
    GO TO 600                         !GO WAIT FOR USER TO RET FROM EVM
C*****
C***** FREQ/PHASE RESPONSE SELECTED - OPTION 2
C*****
C***** 620 CALL WRITEMSG(3,'P3I',ISTAT)      !TURN OFF 172B FOR SURE
C
C READ BANDWIDTH AND CONVERT TO HERTZ
    CALL FDV$CDISP("MENU11")          !FMS MENU 11
    CALL FDV$GETAL(OPP6,TERMINATOR,'CH1')
    CALL SIXCHAR(OPP6,CLBW)            !CONV RET CHAR TO NUMBER
    PI=355/113                        !PI=3.1415929 - TRY IT!!
    CLFR=CLBW/(2*PI)                 !CLFR IS CLOSED-LOOP BW IN HZ
C
C CHECK IT
C
    IF (CLFR .GT. 25600) THEN
        CALL CLEARSCEEN
        PRINT*, 'YOU HAVE EXCEEDED THE LIMITS OF DICES !!!!'
        PRINT*
        PRINT*, 'THE CLOSED-LOOP BANDWIDTH MUST BE LESS THAN '
        PRINT*, '25.6 KHZ.'
        PRINT*
        PRINT*
        CALL WAIT
        GO TO 600
    ENDIF
C

```

```

C*****+
C DISPLAY MESSAGE - ONE MOMENT PLEASE
C
C     CALL FDVACDISP('MENU15')
C
C
C
C*****+
C
CALL FREQSET           !FREQ/PHASE DISPLAYS
CALL WRITEMSG(2,'ED SL,2',ISTAT) !RECALL #2
CALL CONVCHARFS(CLFR,FRC)      !CONVERT TO CHARDATA
CALL WRITEMSG(2,'EM FB',//FRC,ISTAT) ! SET FREQ SPAN
CALL WRITEMSG(2,'EM BG,3',ISTAT) !SET GEN TO RANDOM NOISE
CALL WRITEMSG(2,'DF LM',ISTAT)   !DISPLAY LOWER GRAPH
CALL WRITEMSG(2,'ED SL,3',ISTAT) !RECALL PHASE DISPLAY #3
CALL WRITEMSG(2,'DF UL',ISTAT)   !DISPLAY UPPER & LOWER
CALL WRITEMSG(2,'KP 6',ISTAT)    !TURN ON 'CONT' RECORD
CALL WRITEMSG(2,'CP IF',ISTAT)   !INV FLASHING VIDEO BK
CALL WRITEMSG(2,'PR "PRESS *CONTROL*START* TO BEGIN TEST.
+...',ISTAT)
C
C TEST GENERATOR FLAG TO SEE IF ALREADY ON
IF (BENFLG .NE. 1) THEN          !TOGGLE GEN BUTTON IF OFF
    BENFLG=1
    CALL WRITEMSG(2,'KP [',ISTAT)  ! TURN ON GEN
ENDIF
C
C DISPLAY MESSAGE TO USER
CALL FDVACDISP('MENU16')        !CONTROL T MSG/CTRL START
CALL WAITSTART                 !WAIT FOR *START* SIGNAL
CALL WRITEMSG(2,'PR "
+      "",ISTAT)
CALL WRITEMSG(2,'PR "MEASUREMENT IN PROGRESS...
+      "",ISTAT)
C*****+
C FREQUENCY RESPONSE TEST IN PROGRESS
CALL FDVACDISP('MENU21')         !FREQ RESP TEST IN PROGRESS
C*****+
CALL WAITSTARTOUT               !WAIT FOR START LED TO GO OUT
C
C END OF TEST
C*****+
C SET UP FINAL BK DISPLAY FORMAT AND TURN GENERATORS OFF
C
CALL WRITEMSG(2,'KP [',ISTAT)    !TURN BK GEN OFF
BENFLG=0                         !RESET GEN FLG TO OFF
J=IBUPU(0,2,0)                   !CLEAR BK 2032
J=IBUPU(0,5,2)                   !BK TO LOCAL
CALL WRITEMSG(2,'KP D',ISTAT)    !AUTO SCALE Y AXIS
CALL WRITEMSG(2,'KP S',ISTAT)    !MOVE TO CURSOR FIELD
CALL WRITEMSG(2,'KP S',ISTAT)    !MV CURSOR TO UPPER DISP
CALL WRITEMSG(2,'KP K',ISTAT)    !ALIGN CURSORS ON
CALL WRITEMSG(2,'PR "TEST COMPLETE.",ISTAT)
J=IBUPU(0,5,2)                   !BK TO LOCAL

```

C

```

C RETURN TO PERFORMANCE MENU
C
      GO TO 600
630      GO TO 600
640      GO TO 10
C=====
C
C PRINT PAIR.PRT FILE
C FILE CONTAINS POLE/ZERO PAIRINGS AND COEFFICIENTS
C
700      CALL FDVACDISP('MENU19')           !PRINT RE+ST MENU
        CALL FDVACBETAL(0PP, TERMINATOR, 'CH1')
        CALL CHARTONUM(0PP, IOPT7)
        IF ((IOPT7 .LT. 1) .OR. (IOPT7 .GT. 2)) THEN
          GO TO 700
        ENDIF
        GOTO(710, 720) IOPT7
C PRINT PAIR PRT REPORT TO LINE PRINTER
710      CALL PAIRPAT
        GO TO 10
720      GO TO 10
C
C
C=====
9999      CALL CLEARSCEEN
10000     END
C=====
C
C SUBROUTINE TO CONVERT TO CHARACTER DATA
      SUBROUTINE CONVCHAR(AMP, AMPC)
      CHARACTER=3 AMPC
      IF (AMP .LT. 1) THEN
        AMPC='0.'
        AMPC(3:3)=CHAR(INT(AMP*10)+48)
      ELSE
        PRINT*, 'CONVCHAR SUB - AMP=', AMP
        PRINT*, CHAR(INT(AMP)+48)
        AMPC=CHAR(INT(AMP)+48)//'.'
        PRINT*, 'AMPC (WITH . ADDED) ', AMPC
        AMPC=INT(AMP)
        PRINT*, 'INT((AMP-INT(AMP))*10)=', INT((AMP-AMPC)*10)
        AMPC(3:3)=CHAR(INT((AMP*10.0-INT(AMP)*10.0)+48))
        PRINT*, AMPC
      ENDIF
      RETURN
    END
C
C=====
C SUBROUTINE TO SET BAK INPUT LEVEL
      SUBROUTINE SETBAKINLEVEL(AMP)
      CHARACTER=3 BAKLEV
      CALL CONVCHAR(AMP*2, BAKLEV)
      PRINT*, BAKLEV
      CALL WRITEDB(2, 'BN BN', //BAKLEV, ISTAT) !WRITE TO BAK
      RETURN

```

```

      END
C
C*****SUB TO WAIT FOR START LED TO GO OUT
      SUBROUTINE WAITSTARTOUT
      CHARACTER#7 KEY
      IKEY=0
      DO WHILE (IKEY .EQ. 2)
          J=IBUPU(0, 0, 2, XREF('CS AB' // CHAR(10)), 6)
          J=IBUPU(0, 1, 2, XREF(KEY), 7)
          IKEY=ICHAR(KEY(6:6))-48
          CALL WAITPERIOD(2.0) !WAIT 2 SECONDS
      END DO

      RETURN
      END
C*****SUBROUTINE TO WAIT FOR *START* KEY PRESS
      SUBROUTINE WAITSTART
      CHARACTER#7 KEY
      IKEY=0
      DO WHILE (IKEY .NE. 2)
          J=IBUPU(0, 0, 2, XREF('CS AB' // CHAR(10)), 6)
          J=IBUPU(0, 1, 2, XREF(KEY), 7) !READ LAST KEY
          J=IBUPU(0, 5, 2) !LOCAL
          IKEY=ICHAR(KEY(6:6))-48
          CALL WAITPERIOD(2.0) !WAIT 2 SECONDS
      END DO
      RETURN
      END
C*****SUBROUTINE TO WAIT FOR 'ONTIME' SECONDS
      SUBROUTINE WAITPERIOD(ONTIME)
      T1=SECNDS(0.0)
      DO WHILE (DELTA .LT. ONTIME)
          DELTA=SECNDS(T1)
      END DO
      DELTA=0
      RETURN
      END
C*****SUBROUTINE TO KILL TIME (2 SECONDS)
C
C      SUBROUTINE KILLTIME
C10    T1=SECNDS(0.0)
C      DO WHILE (DELTA .LT. 2.0) ! 2 SECONDS
C          DELTA=SECNDS(T1)
C      END DO
C      DELTA=0
C      RETURN
C      END
C*****SUBROUTINE TO CONVERT TO 5-CHARDATA
      SUBROUTINE CONVCHARFS(CLFR, FRC)

```

```

CHARACTER=5 FRC
IF ((CLFR .LT. 1.56) THEN
  FRC='1.56'
ELSEIF ((CLFR .GT. 1.56) .AND. (CLFR .LE. 3.125)) THEN
  FRC='3.125'
ELSEIF ((CLFR .GT. 3.125) .AND. (CLFR .LE. 6.25)) THEN
  FRC='6.25'
ELSEIF ((CLFR .GT. 6.25) .AND. (CLFR .LE. 12.5)) THEN
  FRC='12.5'
ELSEIF ((CLFR .GT. 12.5) .AND. (CLFR .LE. 25)) THEN
  FRC='25'
ELSEIF ((CLFR .GT. 25) .AND. (CLFR .LE. 50)) THEN
  FRC='50'
ELSEIF ((CLFR .GT. 50) .AND. (CLFR .LE. 100)) THEN
  FRC='100'
ELSEIF ((CLFR .GT. 100) .AND. (CLFR .LE. 200)) THEN
  FRC='200'
ELSEIF ((CLFR .GT. 200) .AND. (CLFR .LE. 400)) THEN
  FRC='400'
ELSEIF ((CLFR .GT. 400) .AND. (CLFR .LE. 800)) THEN V!B3RC='800'
ELSEIF ((CLFR .GT. 800) .AND. (CLFR .LE. 1600)) THEN
  FRC='1600'
ELSEIF ((CLFR .GT. 1600) .AND. (CLFR .LE. 3200)) THEN
  FRC='3200'
ELSEIF ((CLFR .GT. 3200) .AND. (CLFR .LE. 6400)) THEN
  FRC='6400'
ELSEIF ((CLFR .GT. 6400) .AND. (CLFR .LE. 12800)) THEN
  FRC='12800'
ELSEIF (CLFR .GT. 12800) THEN
  FRC='25600'
ENDIF

RETURN
END
C
C*****SUBROUTINE CLS*****
C CLEAR SCREEN SUBROUTINE
C
SUBROUTINE CLS
DO 9998 I=1,24
PRINT#
9998 CONTINUE
RETURN
END
C*****SUBROUTINE CR*****
C PRINT (CR) AND WAIT FOR CARRIAGE RETURN INPUT
C
SUBROUTINE CR
INTEGER DUMMY
PRINT*, '(RETURN) TO CONTINUE'
READ (6,10) DUMMY
10 FORMAT (I)
RETURN
END
C*****SUBROUTINE WAIT*****

```

```

C SUBROUTINE TO INITIALIZE BK AND 1729
C AND DISPLAY BANNER ON BK
C
C
SUBROUTINE INITCONFIG
INTEGER IBUP, IBUPU
CHARACTER#1 LF
CHARACTER#5 BAN
LF=CHAR(10) !DEFINE (LF) CHAR
C
C DISPLAY DICES BANNER FIRST ON BK
CALL WRITEM8B(2,'DF UF',ISTAT) !FULL DISPLAY
CALL WRITEM8B(2,'PF 1,2',ISTAT) !CLR GRAPH AREA
J=IBUPU(0,0,2,XREF ('WT CH," ""//LF),10) !CLR TXT AREA
J=IBUPU(0,0,2,XREF ('PF 60,125,EA,0,0,"DICES"//CHAR(10)),25)
RETURN
END

C
C CLEAR INSTRUMENTS FIRST
C
C SET UP 1729 FOR STEP TEST (NOT USED FOR FREQ/PHASE TEST)
C SET UP DC MODE EACH TIME THROUGH
C STEP TEST. DONE IN STEP TEST SECTION.
C CALL WRITEM8B(3,'D000CAP1I\n',ISTAT) !0 VOLTS DC MODE
C
C
C SET UP BK FOR STEP RESPONSE TEST
C
C
SUBROUTINE STEPSET
J=IBUPU(0,4,2)
CALLWRITEM8B(2,'SR 9',ISTAT) !REMOTE
CALL WRITEM8B(2,'DT '//CHAR(10),ISTAT) !SYS RESET 9
C
CALL WRITEM8B(2,'ED FU,1',ISTAT) !DEF TERMINATOR
CALL WRITEM8B(2,'ED YF,2.00',ISTAT) !CHAR (LF)
CALL WRITEM8B(2,'EM TM,5',ISTAT) !CH B TIME
CALL WRITEM8B(2,'EM BI,0',ISTAT) !2 V FB
CALL WRITEM8B(2,'EM BB,0',ISTAT) !EXT TRIG MODE
CALL WRITEM8B(2,'EM BB,0',ISTAT) !DC DIRECT CH B
CALL WRITEM8B(2,'EM BB,0',ISTAT) !BIEN OFF
C NOW SAVE DISPLAY AND MEAS SETTINGS IN #1
C CALL WRITEM8B(2,'ED 00,1',ISTAT) !DISPLAY STEP TEST
C CALL WRITEM8B(2,'EM 00,1',ISTAT) !MEAS STEP TEST
C
RETURN
END

C
C FREQ/PHASE SETTINGS FOR BK
C
C
SUBROUTINE FREQPSET
CALL WRITEM8B(2,'SR 9',ISTAT) !SYS RESET 9
CALL WRITEM8B(2,'DT '//CHAR(10),ISTAT) !DEF TERMINATOR
C
CALL WRITEM8B(2,'ED SL,12',ISTAT) !CHAR (LF)
CALL WRITEM8B(2,'EM SL,12',ISTAT) !RECALL BK #12 DISP
C
CALL WRITEM8B(2,'EM SL,12',ISTAT) !RECALL BK #12 MEAS

```

```

CALL WRITEMSG(2,'ED XL,1',ISTAT)      !LOS X AXIS
CALL WRITEMSG(2,'EM AI,0',ISTAT)      !CH A DIRECT
CALL WRITEMSG(2,'EM BI,0',ISTAT)      !CH B DIRECT
C
C
C*****SET UP PHASE DISPLAY AND SAVE IN #3
C SET UP PHASE DISPLAY AND SAVE IN #3
C
    CALL WRITEMSG(2,'ED SB,2',ISTAT)  !FREQ TEST (DISP)
C
C NOW CHANGE TO PHASE AND SAVE IN #3
    CALL WRITEMSG(2,'ED FC,3',ISTAT)      !CHB TO PHASE
    CALL WRITEMSG(2,'ED XL,1',ISTAT)      !LOS X AXIS
    CALL WRITEMSG(2,'ED SB,3',ISTAT)      !SAVE IN #3
C
C*****SUBROUTINE TO WRITE OUT TO BOARD 6, FUNCTION 6 (WRITE)
C
SUBROUTINE WRITEMSG(DEV,MSS,IST)
CHARACTER*(*) MSS
CHARACTER*1 LF
INTEGER IBUP,IBUPU,DEV,IST
LF=CHAR(10)
J=IBUPU(6,6,DEV,XREF(MSS//LF),(LEN(MSS)+1))
IST=J
CALL WAITPERIOD(.2)                  !WASTE A LITTLE TIME
RETURN
END
C*****SUBROUTINE TO ASK IF B&K AND 1728 ARE TURNED ON
C
SUBROUTINE CONFIGEQUIP
INTEGER IBUP,IBUPU
C
J=IBUPU(6,2,2)
J=IBUPU(6,4,2)
IF (J .EQ. 1) THEN
    CALL INITCONFIG
ENDIF
RETURN
END
C*****POLE-ZERO PAIRING ALGORITHM
C
C
C*****VARIABLES

```

```

C      COMPLEX
C      #####
C      CPOLE
C      CPOLEPAIR      COMPLEX ARRAY OF POLES (8 MAX)
C      CZERO          REFERENCE POLE TO MEASURE DISTANCE
C      COME           OF ZEROS FROM
C      CORDPZ         COMPLEX ARRAY OF ZEROS (8 MAX)
C                  (1 + j0)
C                  PAIRED POLES AND ZEROS FOR 4 SECTIONS
C                  1.    ZERO
C                  2.    ZERO
C                  3.    POLE   SECTION 1
C                  4.    POLE
C
C                  5.    :
C                  6.    :
C                  .
C                  .
C                  16.   POLE -
C
C      INTEGER
C      #####
C      INCN          NUMBER OF COMPLEX POLES
C      INRP          NUMBER OF REAL POLES
C      ICLOSECPAIR   CLOSEST COMPLEX PAIR INDICES
C      ICLOSERPAIR   CLOSEST REAL PAIR INDICES
C      ICLOSER1      CLOSEST REAL POLE
C      IDORD         ORDER OF DENOMINATOR
C      INORD         ORDER OF NUMERATOR
C      ITNCP         TEMP INCN
C      ITNRP         TEMP INRP
C      INSECTS        NUMBER OF 2ND ORDER SECTIONS (1-4)
C      ISECTORD       SECTION ORDERING VECTOR
C      ICIDX          INDEX OF COMPLEX POLES IN VARIABLE 'CPOLE'
C      IRIIDX         INDEX OF REAL POLES IN 'CPOLE'
C      ICIDXZ         INDEX OF COMPLEX ZEROS IN 'CPOLE'
C      IRIIDXZ       INDEX OF REAL ZEROS IN 'CPOLE'
C      IUSEDZEROS    VECTOR OF INDICES OF AVAIL ZEROS
C      IAVAILREALZ   VECTOR OF AVAIL INDICES IN CZERO
C                  FOR A PARTICULAR TYPE SEARCH (REAL
C                  OR COMPLEX)
C      INUMAVAIL     NUMBER OF AVAIL ZEROS FOR PARTICULAR
C                  TYPE SEARCH
C
C      REAL
C      #####
C      POLEDIST      VECTOR OF DISTANCES OF EACH POLE FROM Z=+1
C      HK             LOOP SENSITIVITY
C
C      ASSUMPTIONS:
C      CZERO AND CPOLE MUST CONTAIN THE POLES AND ZEROS OF THE
C      CONTROLLER IN ANY ORDER, BUT COMPLEX PAIRS MUST STAY ADJACENT
C      TO EACH OTHER.

```

```

C
C
C*****+
C      INITIALIZATIONS
C      SUBROUTINE PAIR(HK, INORD, IDORD, CZERO, CPOLE, CORDPZ,
+     INSECTB, ISECTORD)
C      REAL POLEDIST(8), SECTDIST(4)
C      INTEGER ISECTORD(4), ICIDX(8), IRIDX(8), ICLOSECPAIR(2)
C      INTEGER IAVAILCOMPZ(8), ICIDXZ(8), IRIDXZ(8), ICLOSERPAIR(2)
C      INTEGER IUSEDZERO8(8), INUMAVAIL, IAVAILREALZ(8)
C      COMPLEX CPOLE(8), CZERO(8), CONE, CORDPZ(16)
C      COMPLEX CT1, CT2, CPOLEPAIR(2)
C      CHARACTER*84 TITLE
C      CONE=(1,0)
C      DO 5 I=1,8
C      CORDPZ(I+8)=(9999.0, 9999.0)
C      CORDPZ(I)=(9999.0, 9999.0)
C      IUSEDZERO8(I)=0
5      CONTINUE
C
C      OPEN PRINT FILE
C      OPEN(UNIT=6,FILE='PAIR.PNT',STATUS='NEW')
C      PRINT*, 'INPUT TITLE'
C      READ(6,118) TITLE
C118    FORMAT(A24)
C      WRITE(6,*)
C      WRITE(6,*) '          DICES '
C      WRITE(6,*) ' '
C      WRITE(6,*) '      3D DIGITAL CONTROLLER IMPLEMENTATION'
C      WRITE(6,*) '*****'
C
C      FOR TEST
C      INORD=7
C      IDORD=7
C      CPOLE(1)=(-.2,0)
C      CPOLE(2)=(-.15,.6)
C      CPOLE(3)=(-.15,-.6)
C      CPOLE(4)=(-.05,.7)
C      CPOLE(5)=(-.05,-.7)
C      CPOLE(6)=(-.03,.9)
C      CPOLE(7)=(-.03,-.9)
C      CZERO(1)=(-1.0,0)
C      CZERO(2)=(-.7,.7)
C      CZERO(3)=(-.7,-.7)
C      CZERO(4)=(-.4,.9)
C      CZERO(5)=(-.4,-.9)
C      CZERO(6)=(-.2,.93)
C      CZERO(7)=(-.2,-.93)
C
C      END TEST AREA
C
C
C*****+
C      FIND NUMBER OF REAL AND COMPLEX POLES IN CPOLE
C      AND MAKE INDEX TO CPOLE AND CZERO.

```

```

C
      J=0
      K=0
      DO 10 I=1, IDORD
          IF (AIMAG(CPOLE(I)) .NE. 0) THEN
              J=J+1
              ICIDX(J)=I
              INCP=J
          ELSE
              K=K+1
              IRIDX(K)=I
              INRP=K
          ENDIF
 10      CONTINUE
C*****FIND DISTANCE OF EACH POLE FROM Z=+1 AND PUT IN ARRAY
C      POLEDIST
C
C
      DO 20 I=1, IDORD
          POLEDIST(I)=CABS(CPOLE(I)-CONE)
 20      CONTINUE
C
C*****PAIR UP POLES - CLOSEST AND FARthest REAL POLES FROM Z=+1
C*****PAIRED UP. MAY HAVE A SINGLE FIRST ORDER POLE BY ITSELF.
C*****PUT PAIRS IN ARRAY CORDPZ (HAS 16 ELEMENTS)
C
C
C      REALS FIRST
      ITNRP=INRP
      RR=INRP           !MAKE REAL FOR INTINT FUNCTION
      DO 30 ISECT=1, ININT(RR/2)
          CORDPZ(4*(ISECT-1)+3)=CPOLE(IRIDX(ISECT))
          ITNRP=ITNRP-1
          IF (ITNRP .NE. 0) THEN
              CORDPZ(4*(ISECT-1)+4)=CPOLE(IRIDX(INRP+1-ISECT))
              ITNRP=ITNRP-1
          ELSE
              GO TO 30
          ENDIF
 30      CONTINUE
C
C ****

```

```

C NOW COMPLEX
C RR=INRP      REAL VARIABLE OF NUMBER OF REAL POLES
C
J=1
ITNCP=INCP
C START AT SECTION [ININT((RR/2)+1]
IST=ININT(RR/2)+1
IEND=ININT(RR/2)+(INCP/2)
DO 40 ISECT=IST, IEND
    CORDPZ(4*(ISECT-1)+3)=CPOLE(ICIDX(J))
    CORDPZ(4*(ISECT-1)+4)=CPOLE(ICIDX(J+1))
    J=J+2
40      CONTINUE
C ######
C ######
C FIND CLOSEST POLE TO Z=+1 (REAL OR COMPLEX)
C AND ORDER THEM USING 'ISECTORD'
C RR= REAL VARIABLE - NUMBER OF REAL POLES FOR ININT)
C
INSECTS=(ININT(RR/2))+(INCP/2)
ITNRP=INRP
C
DO 50 I=1,INSECTS
IF (AIMAG(CORDPZ(I))) .EQ. 0) THEN          ! ITS REAL
    CT1=CORDPZ(4*(I-1)+3)
    ITNRP=ITNRP-1
    IF (ITNRP .NE. 0) THEN                      ! TWO REAL POLES
        ITNRP=ITNRP-1
        CT2=CORDPZ(4*(I-1)+4)
        SECTDIST(I)=MIN(CABS(CT1-CONE),CABS(CT2-CONE))
        GO TO 50
    ELSE                                         ! ONLY ONE REAL POLE
        SECTDIST(I)=CABS(CT1-CONE)
    ENDIF
ELSE                                         ! ITS COMPLEX
    CT1=CORDPZ(4*(I-1)+3)
    SECTDIST(I)=CABS(CT1-CONE)
ENDIF
50      CONTINUE
C
C #####
C #####
C NOW ORDER POLES BY CLOSEST TO Z=+1
C (REAL OR COMPLEX)
C PLACE SECTION NUMBER IN 'ISECTORD'
C (WILL HAVE SECTION N IN ISECTORD IN
C LOCATIONS 1-4). CLOSEST SECTION TO
C Z=+1 WILL BE IN LOCATION 1.
C
C
C FILL ISECTORD WITH DEFAULT VALUES TO START
C
DO 60 I=1,4
    ISECTORD(I)=I
60      CONTINUE

```

```

C
C
      DO 70 J=(INSECTS-1),1,-1
      DO 65 I=1,J
         IF (SECTDIST(ISECTORD(I)) .GT.
+           SECTDIST(ISECTORD(I+1))) THEN
            TEMP=ISECTORD(I)
            ISECTORD(I)=ISECTORD(I+1)
            ISECTORD(I+1)=TEMP
         ENDIF
65      CONTINUE
70      CONTINUE
C*****BUILD IUSEDZEROS VECTOR
C
      J=0
      K=0
      DO 6 I=1,INORD
         IF (AIMAG(CZERO(I)) .NE. 0) THEN
            J=J+1
            ICIDXZ(J)=I
            INCZ=J
         ELSE
            K=K+1
            IRIDXZ(K)=I
            INRZ=K
         ENDIF
6      CONTINUE
      DO 7 I=1,J
         IUSEDZEROS(ICIDXZ(I))=ICIDXZ(I)
7      CONTINUE
      DO 8 I=1,K
         IUSEDZEROS(IRIDXZ(I))=IRIDXZ(I)
8      CONTINUE
      WRITE(6,*),'NUMBER OF '
      WRITE(6,*)'SECOND-ORDER SECTIONS',INSECTS
      WRITE(6,*)'NUMERATOR ORDER      ',INORD
      WRITE(6,*)'DENOMINATOR ORDER    ',IDORD
      WRITE(6,*)'LOOP SENSITIVITY     ',MK

C*****MATCH ZEROS TO POLE PAIRS
C
C      (TO PRINT FILE FOR TEST)
      WRITE(6,*)'*****'
      WRITE(6,*)'          UNPAIRED POLES AND ZEROS'
      WRITE(6,*)'*****'
      WRITE(6,*)'CZERO'
      DO 90 M=1,INORD
      WRITE(6,*)'CZERO(M)'
90      CONTINUE
      WRITE(6,*)' '
      WRITE(6,*)'CPOLE'

```

```

DO 99 M=1, IDORD
WRITE(8,*) CPOLE(M)
CONTINUE

99
C
C
C
DO 100 I=1, INSECTS
TT=ISECTORD(I)
WRITE(8,*) '*****'
WRITE(8,*) '      SECOND-ORDER SECTION PAIRINGS'
WRITE(8,*) '*****'
WRITE(8,*) 'SECTION NUMBER   ',I
WRITE(8,*) '*****'
CPOLEPAIR(1)=CORDPZ(4*(TT-1)+3)
CPOLEPAIR(2)=CORDPZ(4*(TT-1)+4)
WRITE(8,*) 'CPOLEPAIR FOR THIS TIME THRU LOOP'
WRITE(8,*) CPOLEPAIR(1)
WRITE(8,*) CPOLEPAIR(2)
WRITE(8,*) ' '
C
IF (REAL(CORDPZ(4*(TT-1)+4)) .EQ. 9999.) THEN !FIRST ORDER
C FIRST ORDER
    CALL ANYREALZEROS(CZERO, IUSEDZEROS, IAVAILREALZ, INUMAVAIL)
    IF (INUMAVAIL .EQ. 0) THEN !NONE AVAIL - DONE
        GO TO 105
    ELSE
        CALL CLOSEST1ZERO(CZERO, CPOLEPAIR, IAVAILREALZ, INUMAVAIL,
        ICLOSER1)
        WRITE(8,*) 'ICLOSER1 = ', ICLOSER1
        WRITE(8,*) CZERO
        CORDPZ(4*(TT-1)+1)=CZERO(ICLOSER1)
        CALL MARK1Z(IUSEDZEROS, ICLOSER1) !MARK USED ZERO
105    ENDIF
C
C 2ND ORDER
C
ELSE
D    WRITE(8,*) '2. MADE IT !!!!!'
    IF (AINAB(CORDPZ(4*(TT-1)+3)) .NE. 0) THEN !COMPLEX POLE
        CALL ANYCOMPZEROS(CZERO, IUSEDZEROS, IAVAILCOMPZ,
        INUMAVAIL)
        IF (INUMAVAIL .EQ. 0) THEN !NONE AVAIL
            CALL ANYREALZEROS(CZERO, IUSEDZEROS, IAVAILREALZ,
            INUMAVAIL)
        D    WRITE(8,*) 'INUMAVAIL= 222 ', INUMAVAIL
            IF (INUMAVAIL .EQ. 0) THEN !NO REALS EITHER
                ENDIF
                IF (INUMAVAIL .EQ. 1) THEN
D        WRITE(8,*) 'CZERO=', CZERO(1) !1 AVAIL
                    CORDPZ(4*(TT-1)+1)=CZERO(IAVAILREALZ(1))
                    CALL MARK1Z(IUSEDZEROS, IAVAILREALZ(1))
                ENDIF
                IF (INUMAVAIL .GT. 1) THEN
                    CALL CLOSESTREALPAIRC(CZERO, CPOLEPAIR,
                    IAVAILREALZ,

```

```

        +
        INUMAVAIL, ICLOSECPAIR)
        CORDPZ(4*(TT-1)+1)=CZERO(ICLOSECPAIR(1))
        CORDPZ(4*(TT-1)+2)=CZERO(ICLOSECPAIR(2))
        CALL MARK2Z(IUSEDZEROS, ICLOSECPAIR)
        ENDIF
    ELSE                                !COMPLEX ZEROS AVAIL
        CALL CLOSESTCOMPLEXPAIR(CZERO, CPOLEPAIR, IAVAILCOMPZ,
        INUMAVAIL, ICLOSECPAIR)
        CORDPZ(4*(TT-1)+1)=CZERO(ICLOSECPAIR(1))
        CORDPZ(4*(TT-1)+2)=CZERO(ICLOSECPAIR(2))
        CALL MARK2Z(IUSEDZEROS, ICLOSECPAIR)
        ENDIF
    ELSE
        D      WRITE(8,*)'MADE IT TO REAL POLES'
            CALL ANYREALZEROS(CZERO, IUSEDZEROS, IAVAILREALZ,
            +
            INUMAVAIL)                                !REAL POLES
        D      WRITE(8,*)'INUMAVAIL=', INUMAVAIL
            IF (INUMAVAIL .EQ. 0) THEN
                IS=INUMAVAIL                      !SAVE # AVAIL
                CALL ANYCOMPZEROS(CZERO, IUSEDZEROS, IAVAILCOMPZ,
                INUMAVAIL)
                IF (INUMAVAIL .EQ. 0) THEN
                    GO TO 100
                ELSE
                    CALL CLOSESTCOMPLEXPAIR(CZERO, CPOLEPAIR,
                    IAVAILCOMPZ, INUMAVAIL, ICLOSECPAIR)
                    CORDPZ(4*(TT-1)+1)=CZERO(ICLOSECPAIR(1))
                    CORDPZ(4*(TT-1)+2)=CZERO(ICLOSECPAIR(2))
                    CALL MARK2Z(IUSEDZEROS, ICLOSECPAIR)
                100   ENDIF
                INUMAVAIL=IS                          !RESTORE # AVAIL
                ENDIF
                IF (INUMAVAIL .EQ. 1) THEN
                    D      WRITE(8,*)'REAL ZERO'
                        IS=INUMAVAIL
                        CALL ANYCOMPZEROS(CZERO, IUSEDZEROS, IAVAILCOMPZ,
                        INUMAVAIL)
                        IF (INUMAVAIL .EQ. 0) THEN
                            CORDPZ(4*(TT-1)+1)=CZERO(IAVILREALZ(1))
                            CALL MARK1Z(IUSEDZEROS, IAVILREALZ(1))
                            GO TO 100
                        ELSE
                            CALL CLOSESTCOMPLEXPAIR(CZERO, CPOLEPAIR,
                            IAVAILCOMPZ, INUMAVAIL, ICLOSECPAIR)
                            CORDPZ(4*(TT-1)+1)=CZERO(ICLOSECPAIR(1))
                            CORDPZ(4*(TT-1)+2)=CZERO(ICLOSECPAIR(2))
                            CALL MARK2Z(IUSEDZEROS, ICLOSECPAIR)
                        100   ENDIF
                        INUMAVAIL=IS
                    ENDIF
                    IF (INUMAVAIL .GT. 1) THEN
                        IS=INUMAVAIL
                        CALL CLOSESTREALPAIR(CZERO, CPOLEPAIR, IAVILREALZ,
                        INUMAVAIL, ICLOSECPAIR)
                        CORDPZ(4*(TT-1)+1)=CZERO(ICLOSECPAIR(1))

```

```

        CORDPZ(4*(TT-1)+2)=CZERO(ICLOSERPAIR(2))
        CALL MARK2Z(IUSEDZEROS, ICLOSERPAIR)
    ENDIF
    ENDIF
    WRITE(6,*) 'CORDPZ ZEROS(1) = ', CORDPZ(4*(TT-1)+1)
    WRITE(6,*) 'CORDPZ ZEROS(2) = ', CORDPZ(4*(TT-1)+2)
    WRITE(6,*) 'CORDPZ POLES(1) = ', CORDPZ(4*(TT-1)+3)
    WRITE(6,*) 'CORDPZ POLES(2) = ', CORDPZ(4*(TT-1)+4)
C     WRITE(6,*) 'END OF MAIN LOOP'
100   CONTINUE
C*****SUBROUTINE ANYCOMPZEROS*****
C
C PRINT OUT CORDPZ ARRAY FOR TEST
D   DO 333 M=1,16
D   WRITE(6,*) CORDPZ(M)
333   CONTINUE
C
C
C     RETURN
END
C*****SUBROUTINE LIBRARY*****
C*****SUBROUTINE ANYCOMPZEROS*****
C
C FUNCTION: DETERMINES IF ANY COMPLEX ZEROS ARE AVAILABLE (EXIST OR
C NOT USED YET IN THE PAIRING PROCESS)
C
C INPUT: CZERO      ARRAY OF ZEROS
C        IUSEDZEROS  VECTOR OF CZERO INDICES OF USED/UNUSED ZEROS
C                      IUSEDZEROS(1)=1   (0 IF NOT AVAIL)
C                      IUSEDZEROS(2)=2
C                      ETC.
C
C OUTPUT: IAVAILCOMPZ  VECTOR OF CZERO INDICES OF ZEROS AVAIL
C        INUMAVAIL    NUMBER OF ZEROS AVAIL
C
C*****SUBROUTINE ANYCOMPZEROS(CZERO, IUSEDZEROS,
+ IAVAILCOMPZ, INUMAVAIL)
COMPLEX CZERO(8)
INTEGER IUSEDZEROS(8), IAVAILCOMPZ(8), INUMAVAIL
DO 5 I=1,8
IAVAILCOMPZ(I)=0
CONTINUE
J=0
DO 10 I=1,8
IF (IUSEDZEROS(I) .NE. 0) THEN
    IF (AIMAG(CZERO(I)) .NE. 0) THEN
        J=J+1

```

```

IAVAILCOMPZ(J)=I
ENDIF
ENDIF
10 CONTINUE
INUMAVAIL=J
RETURN
END
C=====
C=====
C SUBROUTINE ANYREALZROS
C
C FUNCTION: DETERMINES IF ANY REAL ZEROS ARE AVAILABLE
C
C INPUT: CZERO          ZEROS
C        IUSEDZEROS    VECTOR OF ZEROS IN CZERO THAT ARE AVAILABLE
C                      AND OF THOSE USED ALREADY IN A MATCH-UP
C                      WITH A POLE.
C
C OUTPUT:IAVAILREALZ   VECTOR OF INDICES OF AVAIL REAL ZEROS
C
C        INUMAVAIL      NUMBER OF AVAIL ZEROS
C=====
C
SUBROUTINE ANYREALZROS(CZERO, IUSEDZEROS, IAVAILREALZ,
+ INUMAVAIL)
COMPLEX CZERO(8)
INTEGER IUSEDZEROS(8), IAVAILREALZ(8), INUMAVAIL
INUMAVAIL=0
J=0
DO 10 I=1,8
  IF (IUSEDZEROS(I) .NE. 0) THEN           !AVAILABLE
    IF (AIMAG(CZERO(IUSEDZEROS(I))) .EQ. 0) THEN !REAL TOO
      J=J+1
      IAVAILREALZ(J)=IUSEDZEROS(I)
      INUMAVAIL=INUMAVAIL+1
C PRINT FOR TEST ONLY
C     PRINT*, 'I=', I
C     PRINT*, 'J=', J
C     PRINT*, 'INUMAVAIL=', INUMAVAIL
C     PRINT*, 'IUSEDZEROS(I)=' , IUSEDZEROS(I)
C     PRINT*, 'IAVAILREALZ(J)=' , IAVAILREALZ(J)
C END TEST PRINT
      ENDIF
    ENDIF
  10 CONTINUE
  RETURN
END
C=====
C SUBROUTINE MARK1Z
C
C FUNCTION: MARK 1 REAL ZERO IN IUSEDZEROS (WHICH CONTAINS
C INDEX NUMBERS OF AVAIL ZEROS IN CZERO
C
C INPUT: IUSEDZEROS      'VECTOR OF INDICES IN CZERO THAT ARE
C                      'AVAIL

```

```

C      ICLOSER1          ' INDEX # IN CZERO TO MARK
C
C  OUTPUT: IUBEDZEROS      ' MARKED VERSION
C
C*****+
C
C      SUBROUTINE MARK1Z(IUBEDZEROS, ICLOSER1)
C      INTEGER IUBEDZEROS(8), ICLOSER1
C      IUBEDZEROS(ICLOSER1)=0
C      RETURN
C      END
C*****+
C*****+
C      SUBROUTINE MARK2Z
C
C  FUNCTION: MARK 2 REAL ZEROS IN IUBEDZEROS (WHICH CONTAINS
C  INDEX NUMBERS OF AVAIL ZEROS IN CZERO
C
C  INPUT: IUBEDZEROS      ' VECTOR OF INDICES IN CZERO THAT ARE
C          ' AVAIL
C          ICLOSEPAIR OR 'INDEX #8 (2) IN CZERO TO MARK
C          ICLOSECPAIR
C  OUTPUT: IUBEDZEROS      ' MARKED VERSION
C
C*****+
C
C      SUBROUTINE MARK2Z(IUBEDZEROS, ICLOSE)
C      INTEGER IUBEDZEROS(8), ICLOSE(2)
C      IUBEDZEROS(ICLOSE(1))=0
C      IUBEDZEROS(ICLOSE(2))=0
C      RETURN
C      END
C*****+
C*****+
C      SUBROUTINE CLOSESTZERO
C
C  FUNCTION: FINDS CLOSEST SINGLE POLE TO REF POLE
C
C  INPUT: CZERO          ZEROS
C          CPOLEPAIR    REFERENCE POLE
C          IAVAILREALZ  AVAIL ZEROS
C          INUMAVAIL    NUMBER ZEROS AVAIL
C
C  OUTPUT: ICLOSER1      INDEX IN CZERO OF CLOSEST SINGLE ZERO
C*****+
C
C      SUBROUTINE CLOSESTZERO(CZERO, CPOLEPAIR, IAVAILREALZ,
+     INUMAVAIL, ICLOSER1)
C      COMPLEX CZERO(8), CPOLEPAIR(2)
C      INTEGER IAVAILREALZ(8), INUMAVAIL
C      ICLOSER1=IAVAILREALZ(1)      !ASSUME IT'S FIRST ONE TO START
C      DO 10 I=2, INUMAVAIL
C          IF (CRSS(CZERO(IAVAILREALZ(I))-CPOLEPAIR(1))) .LT.
+          CRSS(CZERO(ICLOSER1)-CPOLEPAIR(1))) THEN      !IT'S CLOSER
C              ICLOSER1=IAVAILREALZ(I)

```

```

10      ENDIF
CONTINUE
RETURN
END
C*****
C***** SUBROUTINE CLOSESTREALPAIRC
C
C FUNCTION: FINDS CLOSEST REAL PAIR OF ZEROS TO REF POLE
C (ASSUMES THAT THE REF POLE IS COMPLEX)
C
C INPUT: CZERO      ZEROS
C        CPOLEPAIR   REFERENCE POLE
C        IAVAILREALZ  AVAIL ZEROS
C        INUMAVAIL    NUMBER ZEROS AVAIL
C
C OUTPUT: ICLOSERPAIR INDEX IN CZERO OF CLOSEST ZERO PAIR
C                  (TWO ELEMENT VECTOR OF INDICES)
C
C OTHER VARIABLES:
C        DTD          TEMPORARY VARIABLE
C*****
C
C SUBROUTINE CLOSESTREALPAIRC(CZERO,CPOLEPAIR,IAVAILREALZ,
+     INUMAVAIL,ICLOSERPAIR)
COMPLEX CZERO(8),CPOLEPAIR(2)
INTEGER IAVAILREALZ(8),INUMAVAIL,ICLOSERPAIR(2)

C ORDER ALL THE ZEROS
C
DO 10 J=(INUMAVAIL-1),1,-1
DO 20 I=1,J
      IF (CABS(CZERO(IAVAILREALZ(I))-CPOLEPAIR(1)) .GT.
+          CABS(CZERO(IAVAILREALZ(I+1))-CPOLEPAIR(1))) THEN
      !SWAP
      DTD=IAVAILREALZ(I+1)
      IAVAILREALZ(I+1)=IAVAILREALZ(I)
      IAVAILREALZ(I)=DTD
      ENDIF
20      CONTINUE
10      CONTINUE
C
C NOW SET CLOSEST TWO
C
ICLOSERPAIR(1)=IAVAILREALZ(1)
ICLOSERPAIR(2)=IAVAILREALZ(2)
C
RETURN
END
C*****
C***** SUBROUTINE CLOSESTCOMPLEXPAIR
C
C FUNCTION: FINDS CLOSEST COMPLEX PAIR OF ZEROS TO REF POLE
C
```

```

C (ASSUMES THAT TWO OR MORE ARE AVAILABLE)
C
C INPUT: CZERO      ZEROS
C          CPOLEPAIR   REFERENCE POLE
C          IAVAILCOMPZ  AVAIL COMPLEX ZEROS
C          INUMAVAIL    NUMBER ZEROS AVAIL
C
C OUTPUT: ICLOSECPAIR INDEX IN CZERO OF CLOSEST ZERO PAIR
C                  TWO ELEMENT VECTOR
C
C OTHER VARIABLES:
C          CTPAIR      TEMPORARY
C          DIST        TEMPORARY DISTANCE
C          TDIST       TEMPORARY DISTANCE
C
C*****SUBROUTINE CLOSESTCOMPLEXPAIR(CZERO,CPOLEPAIR,IAVAILCOMPZ,
C+ INUMAVAIL,ICLOSECPAIR)
C          COMPLEX CZERO(8),CPOLEPAIR(2)
C          INTEGER IAVAILREALZ(8),INUMAVAIL,ICLOSECPAIR(2),
C+ IAVAILCOMPZ(8)
C
C
C          TDIST=99999           !SET DISTANCE TO 9
C          ICLOSECPAIR(1)=0
C          ICLOSECPAIR(2)=0
C
C          DO 10 I=1,INUMAVAIL,2
C              DIST=CABS(CZERO(IAVAILCOMPZ(I))-CPOLEPAIR(1))
C              IF (DIST .LT. TDIST) THEN           !REPLACE
C                  TDIST=DIST
C                  ICLOSECPAIR(1)=IAVAILCOMPZ(I)
C                  ICLOSECPAIR(2)=IAVAILCOMPZ(I+1)
C              ENDIF
C 10      CONTINUE
C
C          RETURN
C
C*****SUBROUTINE CLOSESTREALPAIRR
C
C FUNCTION: FINDS CLOSEST REAL PAIR OF ZEROS TO REF POLE
C (ASSUMES THAT THE REF POLES ARE REAL)
C
C INPUT: CZERO      ZEROS
C          CPOLEPAIR   REFERENCE POLE
C          IAVAILREALZ  AVAIL ZEROS
C          INUMAVAIL    NUMBER ZEROS AVAIL
C
C OUTPUT: ICLOSERPAIR INDEX IN CZERO OF CLOSEST ZERO PAIR
C                  (TWO ELEMENT VECTOR OF INDICES)
C

```

```

C OTHER VARIABLES:
C      DTD      TEMPORARY VARIABLE
C*****
C
C      SUBROUTINE CLOSESTREALPAIR(CZERO,CPOLEPAIR,IAVILREALZ,
+     INUMAVAIL,ICLOSERPAIR)
COMPLEX CZERO(8),CPOLEPAIR(2)
INTEGER IAVILREALZ(8),INUMAVAIL,ICLOSERPAIR(2)

C
C ORDER ALL THE ZEROS
C
      DO 10 J=(INUMAVAIL-1),1,-1
      DO 20 I=1,J
         IF (CABS(CZERO(IAVILREALZ(I))-CPOLEPAIR(1)) .GT.
+           CABS(CZERO(IAVILREALZ(I+1))-CPOLEPAIR(1))) THEN
               !SWAP
               DTD=IAVILREALZ(I+1)
               IAVILREALZ(I+1)=IAVILREALZ(I)
               IAVILREALZ(I)=DTD
            ENDIF
20      CONTINUE
10      CONTINUE
C
C NOW GET CLOSEST TWO
C
      ICLOSERPAIR(1)=IAVILREALZ(1)
      IF (INUMAVAIL .EQ. 2) THEN
         ICLOSERPAIR(2)=IAVILREALZ(2)
         GO TO 15
      ELSE
         DO 30 J=(INUMAVAIL-1),2,-1
         DO 40 I=1,J
            IF (CABS(CZERO(IAVILREALZ(I))-CPOLEPAIR(2)) .GT.
+              CABS(CZERO(IAVILREALZ(I+1))-CPOLEPAIR(2))) THEN
               !SWAP
               DTD=IAVILREALZ(I+1)
               IAVILREALZ(I+1)=IAVILREALZ(I)
               IAVILREALZ(I)=DTD
            ENDIF
30      CONTINUE
40      CONTINUE
      ICLOSERPAIR(2)=IAVILREALZ(2)
15      ENDIF
C
C
      RETURN
END
C*****
C      PROGRAM READHTF
C      COMPLEX CPOLE(8),CZERO(8)
C
C      INORD=4
C      IDORD=4
C      CALL RDORDER(INORD, IDORD, MK)

```

```

C      PRINT*, INORD, IDORD, HK
C      CALL RDHTF(INORD, IDORD, CZERO, CPOLE)
C      PRINT*, CZERO
C      PRINT*, CPOLE
C      STOP
C      END
C
C*****+
C*****+
C      MODULE NAME: RDHTF.FOR
C      FUNCTION: READS AN ASCII DATA FILE, CHANGES ITS CONTENTS
C      AND THEN REWRITES THE FILE TO A NEW FILE.
C
C      INPUT: S      MAXIMUM ORDER DICES WILL HANDLE
C              INORD   ORDER OF NUMERATOR OF CONTROLLER
C              IDORD   ORDER OF DENOMINATOR OF CONTROLLER
C
C      OUTPUT: CZERO  COMPLEX ARRAY OF ZEROS OF CONTROLLER
C                  CPOLE   COMPLEX ARRAY OF POLES OF CONTROLLER
C
C      OTHER: TRZ    TEMP STORAGE OF REAL PARTS OF ZEROS
C              TIZ    TEMP STORAGE OF IMAG PARTS OF ZEROS
C              TRP    TEMP STORAGE OF REAL PARTS OF POLES
C              TIP    TEMP STORAGE OF IMAG PARTS OF POLES
C              PB     DUMMY VAR FOR 6 DATA ITEMS/LINE
C              RR     DUMMY VAR FOR 2 DATA ITEMS/LINE
C              R4     DUMMY VAR FOR 4 DATA ITEMS/LINE
C*****+
C*****+
C
C      SUBROUTINE RDHTF(INORD, IDORD, CZERO, CPOLE)
REAL LINE(5)
REAL PB(6), RR(2)
REAL R4(4), TRZ(8), TRP(8), TIZ(8), TIP(8)
REAL TT1, TT2
COMPLEX CZERO(8), CPOLE(8)
C
C
C      DO 5 I=1,8
CZERO(I)=(9999, 9999)
CPOLE(I)=(9999, 9999)
5      CONTINUE
C
C*****+ OPEN FILES ++++++
C
C      'MEMORY.DAT' IS THE VARIABLE STORAGE FOR ICECAP.
C      IT IS READ IN, MODIFIED, AND TRANSFERRED
C      TO 'MEMORYNEW.DAT'.
C
OPEN(UNIT=1,FILE='MEMORY.DAT',STATUS='OLD')
REWIND 1
C
C*****+
C

```

```

C      TRANSFER LINES 1-XXX OF ORIGINAL FILE
C
C      READ(1,*)
C      DO 2 I=1,140
C      READ(1,*)
C      K=I+1
C
2      CONTINUE
C      READ(1,*)
C      READ(1,*)
C      READ(1,*)
C      K=K+3
C      DO 4 I=1,120
C      READ(1,*)
C
4      CONTINUE
C      K=K+I-1
C      PRINT*, '***** LINE ',K
C
C*****+
C
C      READ(1,*)
C      READ(1,*)
C      READ(1,*)
C      PRINT*, '*****',K+3
C      PRINT*, ' RR ',RR
C
C      DO 6 I=1,61
C      READ(1,*)
C      CONTINUE
C      PRINT*, 'LINE 328 = ',LINE
C
C      READ(1,*)
C      PRINT*, 'LINE 329 = ',RR
C
C      READ PAST BTF
C      DO 7 I=1,20
C      READ(1,*)
C      PRINT*,LINE
C
7      CONTINUE
C
C*****+
C      READ IN HTF NOW
C      DATA STARTS AT DATA ITEM #3.
C      # OF LINES TO READ IS 1+INT((INORD-3)/5+.8)
C      EX. - IF INORD = 14 THEN LINES READ MUST BE
C      1 FOR FIRST 3 DATA ITEMS
C      3 FOR NEXT 11 ITEMS INT(14-3)/5+.8)=3
C
C      -----
C      4 LINES
C
C      350    X      X      D1      D2      D3
C      D4      D5      D6      D7      D8
C      D9      D10     D11     D12     D13
C      D14     X       X       X       X
C
C*****+

```

```

ITNORD=INORD
READ(1,*) LINE
IF (ITNORD .LE. 3) THEN !JUST 3 OR LESS
    DO 10 I=1,ITNORD
        TRZ(I)=LINE(I+2)           !ZERO REAL PART
10     CONTINUE
ELSE
    DO 20 I=1,3                 !MORE THAN 3
        TRZ(I)=LINE(I+2)         !READ FIRST 3
20     CONTINUE
ITNORD=ITNORD-3             !FOR 3 JUST READ IN
TT3=INORD
DO 30 I=1,INT((TT3-3)/5+.8) !NOW THE REST
    READ(1,*) LINE
    TRZ(3+5*(I-1)+1)=LINE(1) !READ IN TO TEMP VARIABLE
    ITNORD=ITNORD-1
C SECOND ONE IN ROW
    IF (ITNORD .NE. 0) THEN !READ IN ANOTHER
        TRZ(3+5*(I-1)+2)=LINE(2)
        ITNORD=ITNORD-1
    ENDIF
    IF (ITNORD .NE. 0) THEN !READ IN ANOTHER
        TRZ(3+5*(I-1)+3)=LINE(3)
        ITNORD=ITNORD-1
    ENDIF
    IF (ITNORD .NE. 0) THEN !READ IN ANOTHER
        TRZ(3+5*(I-1)+4)=LINE(4)
        ITNORD=ITNORD-1
    ENDIF
    IF (ITNORD .NE. 0) THEN !READ IN ANOTHER
        TRZ(3+5*(I-1)+5)=LINE(5)
        ITNORD=ITNORD-1
    ENDIF
30     CONTINUE               !GO READ ANOTHER ROW
ENDIF
C NOTE: THIS ROUTINE WILL ONLY WORK FOR ORDER 40 BECAUSE
C OF THE WAY THE MEMORY FILE IS SET UP. TO WORK FOR 50, IT
C MUST RE-USE THE LAST LINE READ BECAUSE THERE IS ALSO HTF
C IMAGINARY DATA IN THE LAST LINE.
C
C NOW READ IN ENOUGH LINES TO MOVE TO IMAGINARY PART OF HTF ZEROS.
C THE IMAGINARY DATA STARTS AT LINE 360 (DATA ITEM #3) OF THE
C MEMORY.DAT FILE.
    DO 35 I=1,(10-(1+INT((TT3-3)/5+.8)))
    READ(1,*) LINE
35     CONTINUE
C *****
C NOW READ IN THE IMAGINARY PART OF HTF ZEROS INTO TIZ (TEMP)
    ITNORD=INORD
    READ(1,*) LINE
    IF (ITNORD .LE. 3) THEN !JUST 3 OR LESS
        DO 50 I=1,ITNORD
            TIZ(I)=LINE(I+2)           !ZERO REAL PART
50     CONTINUE
ELSE
    !MORE THAN 3

```

```

DO 60 I=1,3           !READ FIRST 3
TIZ(I)=LINE(I+2)
60      CONTINUE
ITNORD=ITNORD-3
DO 70 I=1, INT((TT3-3)/5+.8)   !FOR 3 JUST READ IN
READ(1,*) LINE          !NOW THE REST
TIZ(3+5*(I-1)+1)=LINE(1)    !READ IN TO TEMP VAR
ITNORD=ITNORD-1

C SECOND ONE IN ROW
IF (ITNORD .NE. 0) THEN    !READ IN ANOTHER
TIZ(3+5*(I-1)+2)=LINE(2)
ITNORD=ITNORD-1
ENDIF
IF (ITNORD .NE. 0) THEN    !READ IN ANOTHER
TIZ(3+5*(I-1)+3)=LINE(3)
ITNORD=ITNORD-1
ENDIF
IF (ITNORD .NE. 0) THEN    !READ IN ANOTHER
TIZ(3+5*(I-1)+4)=LINE(4)
ITNORD=ITNORD-1
ENDIF
IF (ITNORD .NE. 0) THEN    !READ IN ANOTHER
TIZ(3+5*(I-1)+5)=LINE(5)
ITNORD=ITNORD-1
ENDIF
70      CONTINUE           !GO READ ANOTHER ROW
ENDIF

C
C NOW READ IN ENOUGH LINES TO MOVE TO REAL PART OF HTF POLES.
C THE REAL DATA STARTS AT LINE 370 (DATA ITEM #3) OF THE
C MEMORY.DAT FILE.
DO 80 I=1, (10-(1+INT((TT3-3)/5+.8)))
READ(1,*) LINE
80      CONTINUE
*****  

C NOW DO POLES OF HTF
C
C NOW READ IN THE REAL PART OF HTF POLES INTO TRP (TEMP)
ITDORD=IDORD           !TEMP # OF POLES
TT3=IDORD
READ(1,*) LINE
IF (ITDORD .LE. 3) THEN !JUST 3 OR LESS
DO 90 I=1, ITDORD       !REAL PART OF ZERO
TRP(I)=LINE(I+2)
90      CONTINUE
ELSE                     !MORE THAN 3
DO 100 I=1,3             !READ FIRST 3
TRP(I)=LINE(I+2)
100     CONTINUE
PRINT*, 'TRP ', TRP(I)
C
100     CONTINUE
ITDORD=ITDORD-3         !FOR 3 JUST READ IN
DO 110 I=1, INT((TT3-3)/5+.8)   !NOW THE REST
READ(1,*) LINE
TRP(3+5*(I-1)+1)=LINE(1)    !READ IN TO TEMP VAR
ITDORD=ITDORD-1

```

```

C SECOND ONE IN ROW
    IF (ITDORD .NE. 0) THEN      !READ IN ANOTHER
        TRP(3+5*(I-1)+2)=LINE(2)
        ITDORD=ITDORD-1
    ENDIF
    IF (ITDORD .NE. 0) THEN      !READ IN ANOTHER
        TRP(3+5*(I-1)+3)=LINE(3)
        ITDORD=ITDORD-1
    ENDIF
    IF (ITDORD .NE. 0) THEN      !READ IN ANOTHER
        TRP(3+5*(I-1)+4)=LINE(4)
        ITDORD=ITDORD-1
    ENDIF
    IF (ITDORD .NE. 0) THEN      !READ IN ANOTHER
        TRP(3+5*(I-1)+5)=LINE(5)
        ITDORD=ITDORD-1
    ENDIF
110    CONTINUE                  !GO READ ANOTHER ROW
    ENDIF
C     PRINT*, 'REAL PART POLES', TRP
C
C
C NOW READ IN ENOUGH LINES TO MOVE TO IMAG PART OF HTF POLES.
C THE IMAG DATA STARTS AT LINE 300 (DATA ITEM #3) OF THE
C MEMORY.DAT FILE.
    DO 120 I=1,(10-(1+INT((TT3-3)/5+.8)))
    READ(1,*) LINE
120    CONTINUE
C
C NOW READ IN THE IMAG PART OF HTF POLES INTO TIP (TEMP)
    ITDORD=IDORD
    READ(1,*) LINE
    IF (ITDORD .LE. 3) THEN !JUST 3 OR LESS
        DO 130 I=1,ITDORD
        TIP(I)=LINE(I+2)          !ZERO REAL PART
    130    CONTINUE
    ELSE                      !MORE THAN 3
        DO 140 I=1,3            !READ FIRST 3
        TIP(I)=LINE(I+2)
    140    CONTINUE
    ITDORD=ITDORD-3           !FOR 3 JUST READ IN
    DO 145 I=1,INT((TT3-3)/5+.8) !NOW THE REST
    READ(1,*) LINE
    TIP(3+5*(I-1)+1)=LINE(1)   !READ IN TO TEMP VAR
    ITDORD=ITDORD-1
C SECOND ONE IN ROW
    IF (ITDORD .NE. 0) THEN      !READ IN ANOTHER
        TIP(3+5*(I-1)+2)=LINE(2)
        ITDORD=ITDORD-1
    ENDIF
    IF (ITDORD .NE. 0) THEN      !READ IN ANOTHER
        TIP(3+5*(I-1)+3)=LINE(3)
        ITDORD=ITDORD-1
    ENDIF
    IF (ITDORD .NE. 0) THEN      !READ IN ANOTHER

```

```

        TIP(3+5*(I-1)+4)=LINE(4)
        ITDORD=ITDORD-1
    ENDIF
    IF (ITDORD .NE. 0) THEN      !READ IN ANOTHER
        TIP(3+5*(I-1)+5)=LINE(5)
        ITDORD=ITDORD-1
    ENDIF
145    CONTINUE                      !GO READ ANOTHER ROW
    ENDIF
C*****+
C
C NOW FILL UP CPOLE AND CZERO COMPLEX ARRAYS
C
C DO ZEROS
C
        DO 150 I=1,INORD
        TT1=TRZ(I)
        TT2=TIZ(I)
        CZERO(I)=CMPLX(TT1,TT2)
150    CONTINUE
C
C DO POLES
C
        DO 160 I=1, IDORD
        TT1=TRP(I)
        TT2=TIP(I)
        CPOLE(I)=CMPLX(TT1,TT2)
160    CONTINUE
C*****+
C
        RETURN
    END
C
C PROGRAM RD
C
C CALL RDOORDER(INORD, IDORD, HK, TSAMP)
C
C PRINT*, INORD, IDORD, HK, TSAMP
C
C STOP
C
C END
C*****+
C*****+
C
C MODULE NAME: RDOORDER.FOR
C
C FUNCTION: READS THE ORDER OF THE NUMERATOR AND
C
C DENOMINATOR OF HTF.
C
C
C OUTPUT: INORD           ORDER OF NUMERATOR OF CONTROLLER
C          IDORD           ORDER OF DENOM OF CONTROLLER
C          TSAMP            SAMPLE RATE USED IN ICECAP
C
C
C OTHER:
C          PG              DUMMY VAR FOR 6 DATA ITEMS/LINE
C          RR              DUMMY VAR FOR 2 DATA ITEMS/LINE
C          R3              DUMMY VAR FOR 3 DATA ITEMS/LINE
C          R4              DUMMY VAR FOR 4 DATA ITEMS/LINE
C*****+
C*****+
C
C SUBROUTINE RDOORDER(INORD, IDORD, HK, TSAMP)
C
C REAL LINE(5),PG(6),RR(2),R4(4),HK,R3(3)

```

```

C
C*****OPEN FILES *****
C   'MEMORY.DAT' IS THE VARIABLE STORAGE FOR ICECAP.
C
C      OPEN(UNIT=1,FILE='MEMORY.DAT',STATUS='OLD')
C      REWIND 1
C
C***** *****
C
C      READ PAST LINES 1-369 OF FILE
C
C      READ(1,*) P          !READ THE 969 AT FRONT OF FILE
C      DO 2 I=1,140         !READ LINES 2-141 (5 #8 PER)
C      READ(1,*) LINE
C      K=I+1
C
2    CONTINUE
C      READ(1,*) PB         !READ 6 #8
C      READ(1,*) PB         !READ 6 #8
C      READ(1,*) RR         !READ 2 #8 (LINE 144)
C      K=K+3
C      DO 4 I=1,120         !READ LINES 145-264 (5 #8)
C      READ(1,*) LINE
C
4    CONTINUE
C      K=K+I-1
C
C      PRINT*, '***** LINE ',K
C
C***** *****
C
C      READ(1,*) R4          !LINE 265 -4 #8
C      READ(1,*) PB          !LINE 266 -6 #8
C      READ(1,*) RR          !LINE 267 -2 #8
C      PRINT*, '*****',K+3
C      PRINT*, ' RR ',RR
C
C      DO 6 I=1,61           !LINES 268-328
C      READ(1,*) LINE
C
6    CONTINUE
C      PRINT*, 'LINE 328 = ',LINE
C
C      READ(1,*) RR          !LINE 329 -2 #8
C      PRINT*, 'LINE 329 = ',RR
C
C      READ PAST BTF
C      DO 7 I=1,60           !LINES 330-369
C      READ(1,*) LINE
C      PRINT*,LINE
C
7    CONTINUE
C
C***** *****
C      READ IN HTF NOW
C      DATA STARTS AT DATA ITEM #3.
C
C      390 X     X     NUM     DEN     HK
C          HNK     HDK     X     X     X

```

```

C NOW READ THE LINE WITH ORDER IN IT LINE # 390
C
READ(1,*) LINE
INORD=LINE(3)
IDORD=LINE(4)

C NOW COMPUTE HK=HNK/HDK
C AND SET TO TBAMP AT LINE 615(4)
C

READ(1,*) R3                                !LINE 391
HNK=R3(1)                                     !GET HNK
HDK=R3(2)                                     !GET HDK
HK=HNK/HDK                                    !COMPUTE HK
DO 390 I=1,61                                  !LINES 392-452
READ(1,*) LINE
CONTINUE
READ(1,*) RR                                !LINE 453
DO 310 I=1,61                                  !LINES 454-514
READ(1,*) LINE
CONTINUE
READ(1,*) RR                                !515
READ(1,*) PG                                !516
READ(1,*) R3                                !517
DO 320 I=1,27                                  !LINES 518-544
READ(1,*) PG
CONTINUE
DO 330 I=1,4                                   !LINES 545-548
READ(1,*) LINE
CONTINUE
READ(1,*) R3                                !549
DO 340 I=1,30                                  !LINES 550-579
READ(1,*) LINE
CONTINUE
READ(1,*) R3                                !580
DO 350 I=1,32                                  !LINES 581-612
READ(1,*) LINE
CONTINUE
READ(1,*) R4                                !613
READ(1,*) PG                                !614
READ(1,*) R4                                !FINALLY 615

C GET THE TBAMP VALUE
C
TBAMP=R4(4)

C CLOSE (UNIT=1)
C
RETURN
END

```

C NAME: MULTRTB
 C
 C FUNCTION: MULTIPLY OUT THE ROOTS OF THE NUMERATOR
 C AND DENOMINATOR TO YIELD FIRST AND
 C SECOND ORDER POLYNOMIALS AND THEN QUANTIZE
 C THE COEFFICIENTS
 C
 C INPUT: CORDPZ ARRAY OF ZEROS AND POLES FOR EACH
 C SECTION OF FILTER. 9999 MEANS NO
 C ENTRY IN THAT POSITION.
 C
 C NUMSECTS NUMBER OF SECTIONS
 C
 C TSAMP SAMPLE PERIOD FROM ICECAP
 C
 C HK NUMERATOR LOOP SENSITIVITY TERM
 C HIGHEST POWER OF Z COEFFICIENT IN
 C NUMERATOR DIVIDED BY HIGHEST POWER OF Z IN
 C DENOMINATOR.
 C
 C ISECTORD ARRAY CONTAINING ORDER OF SECTIONS
 C
 C TRFLB TRUNCATE OR ROUND-OFF FLAG
 C 1 = TRUNCATE, 0=ROUND-OFF
 C
 C OUTPUT: NEWCO.TXT DATA FILE WITH 21 ELEMENTS THAT
 C CONTAINS THE QUANTIZED COEFFICIENTS
 C IN INTEGER FORM. TSAMP IS AT POSITION 21.
 C FOUR SECTIONS OF FIVE COEFFICIENTS FOR
 C EACH SECTION FOR THE POLYNOMIAL
 C COEFFICIENTS. ORDER IS B0,B1,B2,A1,A2.
 C IF A COEFFICIENT IS ZERO, IT WILL BE A
 C ZERO IN NEWCO. DEFAULT VALUES ARE 32767
 C FOR B0 TERMS. ZERO FOR ALL OTHERS.
 C
 C OTHER: ICNT ORDER OF EACH SECTION (0,1, OR 2)
 C
 C FRACHK FRACTION OF HK SPREAD OVER EACH SECTION
 C THAT IS USED
 C
 C ADJ CORRECTION FOR LOSS OF BIAS BECAUSE OF '1'
 C BEING .999969
 C
 C ITSAMPI INTEGER VERSION OF TSAMP
 C
 C B0, B1, B2 SECTION COEFFICIENTS OF FORM
 C A0, A1
 C $B0 + B1z^{-1} + B2z^{-2}$
 C $G(z) = \frac{B0 + B1z^{-1} + B2z^{-2}}{1 - A0z^{-1} - A1z^{-2}}$
 C ****=
 C SUBROUTINE MULTRTB(CORDPZ,NUMSECTS,TSAMP,HK,ISECTORD,TRFLB)
 C INTEGER INEWCO(21),ISECTORD(4)
 C COMPLEX CORDPZ(16)

```

REAL*8 TT, HK
REAL*8 ADJ, FRACHK, B0, B1, B2, A1, A2
C
C INITIALIZE INEWCO (32767 FOR B0 TERMS, 0 FOR REST)
C THIS REPRESENTS APPROX ONE (HIGHEST +Q15 NUMBER)
C LOSS OF GAIN WILL BE COMPENSATED FOR IN HK THAT GETS
C MULTIPLIED THROUGH THE NUMERATOR.
C
DO 5 I=1,21
INEWCO(I)=0
5 CONTINUE
C
INEWCO(1)=32767
INEWCO(6)=32767
INEWCO(11)=32767
INEWCO(16)=32767
C
C OPEN A PRINT FILE FOR TESTING
C
C
C WRITE(8,*) 'CORDPZ'
C DO 7 I=1,16
C WRITE(8,*) CORDPZ(I)
C7 CONTINUE
C WRITE(8,*),'
C WRITE(8,*),'
C
C GO THROUGH LOOP NUMBER OF SECTIONS USED (NUMSECTS)
C
C
C CORRECTIONS FOR GAIN LOSSES DUE TO .9999694 COEFFICIENTS
C OF SECTIONS THAT ARE NOT USED. (AS CLOSE TO 1 AS CAN GET).
C
C THE TEST FOR 0 IS TO AVOID A DIVIDE BY ZERO AND TO ALLOW ONE
C PASS THROUGH THELOOP THAT COMPUTES THE QUANTIZED COEFFICIENTS
C
C
IF (NUMSECTS .EQ. 0) THEN
FRACHK=1.00012
NUMSECTS=1
ADJ=1.000030519
ELSE
ADJ=(1.000030519**((4-NUMSECTS)**(1.0/NUMSECTS))
FRACHK=HK**((1.0/NUMSECTS))
ENDIF
C
C
WRITE(8,*) ' COEFFICIENT QUANTIZATION '
WRITE(8,*) '*****'
WRITE(8,*) 'NUMSECTS =',NUMSECTS
C WRITE(8,*) 'ADJ =',ADJ
C WRITE(8,*) 'HK =',HK
C WRITE(8,*) 'FRACHK =',FRACHK
C WRITE(8,*) 'TBAMP =',TBAMP
C

```

```

C
DO 10 K=1,NUMBECTS
I=ISECTORID(K)
WRITE(8,*)
C*****+
C NUMERATOR FIRST*
C*****
WRITE(8,*)
WRITE(8,*)
ICNT=0
IF (REAL(CORDPZ(4*(I-1)+1)) .NE. 9999.0) THEN
  ICNT=ICNT+1 !1ST ORDER
ENDIF
IF (REAL(CORDPZ(4*(I-1)+2)) .NE. 9999.) THEN
  ICNT=ICNT+1 !2ND ORDER
ENDIF
C
WRITE(8,*)
C*****
IF (ICNT .EQ. 2) THEN
  !2ND ORDER
  B1=REAL(CORDPZ(4*(I-1)+1))+REAL(CORDPZ(4*(I-1)+2))
  B2=CORDPZ(4*(I-1)+1)*CORDPZ(4*(I-1)+2)
  B1=-ADJ*FRACHK*B1
  B2=ADJ*FRACHK*B2
  B0=FRACHK*ADJ
  WRITE(8,*)
  WRITE(8,*)
  WRITE(8,*)
C
IF (TRFLB .EQ. 1 ) THEN
  IB0=INT(B0+32768)
  IB1=INT(B1+32768)
  IB2=INT(B2+32768) !TRUNCATE
ELSE
  IB0=INT(B0+32768+.5) !ROUND
  IB1=INT(B1+32768+.5)
  IB2=INT(B2+32768+.5)
ENDIF
WRITE(8,*)
WRITE(8,*)
WRITE(8,*)
C
C SAVE COEFFS IN INEWCO
C
INEWCO(5*(I-1)+1)=IB0
INEWCO(5*(I-1)+2)=IB1
INEWCO(5*(I-1)+3)=IB2
ENDIF
C*****+
C FIRST ORDER
C
IF (ICNT .EQ. 1) THEN !1ST ORDER
  IF (REAL(CORDPZ(4*(I-1)+1)) .NE. 9999) THEN !ITS USED
    B1=REAL(CORDPZ(4*(I-1)+1))
  ENDIF
  IF (REAL(CORDPZ(4*(I-1)+2)) .NE. 9999) THEN !ITS USED

```

```

      B1=REAL(CORDPZ(4*(I-1)+2))
      ENDIF
      B0=-B1*ADJ+FRACHK
      B0=FRACHK*ADJ
      WRITE(8,*)'B0 =',B0
      WRITE(8,*)'B1 =',B1
C*****
C TRUNCATE OR ROUND
C
      IF (TRFLB .EQ. 1) THEN
          IB0=INT(B0*32768)
          IB1=INT(B1*32768)
      ELSE
          IB0=INT(B0*32768+.5)
          IB1=INT(B1*32768+.5)
      ENDIF
      WRITE(8,*)'IB0 =',IB0
      WRITE(8,*)'IB1 =',IB1
C*****
C STORE IN INEWCO
C
      INEWCO(5*(I-1)+1)=IB0
      INEWCO(5*(I-1)+2)=IB1
      ENDIF
C
      !0TH ORDER - DO NOTHING
C
C*****
C NOW DENOMINATOR *
C*****
ICNT=0
      IF (REAL(CORDPZ(4*(I-1)+3)) .NE. 9999.9) THEN
          ICNT=ICNT+1 !1ST ORDER
      ENDIF
      IF (REAL(CORDPZ(4*(I-1)+4)) .NE. 9999.9) THEN
          ICNT=ICNT+1 !2ND ORDER
      ENDIF
C
      IF (ICNT .EQ. 2) THEN !2ND ORDER
          A1=REAL(CORDPZ(4*(I-1)+3))+REAL(CORDPZ(4*(I-1)+3))
          A2=-CORDPZ(4*(I-1)+3)*CORDPZ(4*(I-1)+4)
          WRITE(8,*)'A1 =',A1
          WRITE(8,*)'A2 =',A2
C*****
          IF (TRFLB .EQ. 1) THEN
              IA1=INT(A1*32768)
              IA2=INT(A2*32768) !TRUNCATE
          ELSE
              IA1=INT(A1*32768+.5)
              IA2=INT(A2*32768+.5) !ROUND
          ENDIF
          WRITE(8,*)'IA1 =',IA1
          WRITE(8,*)'IA2 =',IA2
C*****
C SAVE COEFFS IN INEWCO

```

```

C
    INEWCO(5*(I-1)+4)=IA1
    INEWCO(5*(I-1)+5)=IA2
    ENDIF
C *****
    IF (ICNT .EQ. 1) THEN          !1ST ORDER
        IF (REAL(CORDPZ(4*(I-1)+3)) .NE. 9999) THEN
            !IS IT 9999?
            A1=REAL(CORDPZ(4*(I-1)+3))
        ENDIF
        IF (REAL(CORDPZ(4*(I-1)+4)) .NE. 9999) THEN
            !IS IT 9999?
            A1=REAL(CORDPZ(4*(I-1)+4))
        ENDIF
        WRITE(8,*), 'A1 =', A1
C *****
C TRUNCATE OR ROUND
C
    IF (TRFLG .EQ. 1) THEN
        IA1=INT(A1*32768)
    ELSE
        IA1=INT(A1*32768+.5)
    ENDIF
    WRITE(8,*), 'IA1 =', IA1
C *****
C STORE IN INEWCO
C
    INEWCO(5*(I-1)+4)=IA1
    ENDIF
C                                     !8TH ORDER - DO NOTHING
10    CONTINUE
C *****
C NOW PUT TSAMP IN POSITION 21 AS AN INTEGER
C AND CHECK IT FOR LIMITS
C
    IF (TSAMP .LT. .BE-3) THEN      !TOO SMALL
C PUT ERROR ROUTINE HERE
C
    ENDIF
    IF (TSAMP .GT. 32.4289) THEN   !TOO LARBE
C
C HERE TOO
    ENDIF
    ITSAMPI=INT((.0524289/(TSAMP/400))+.5)
C                                     !MULT BY 400 IN TM8320 CODE
    WRITE(8,*), '*****'
    WRITE(8,*), 'TSAMP =', TSAMP
    WRITE(8,*), 'ITSAMP =', ITSAMPI
    WRITE(8,*), '*****'
    INEWCO(21)=ITSAMP
C *****
C WRITE OUT INEWCO TO NEWCO.TXT
C

```

```

OPEN(UNIT=5,FILE='NEWCO.TXT',STATUS='NEW')
DO 20 I=1,21
WRITE(5,45) INEWCO(I)
20  CONTINUE
45  FORMAT(1X,I6)
C*****
CLOSE(5)
CLOSE(6)          !CLOSE NEWCO.TXT
RETURN
END

C*****
C
C*****
C
C      MODULE NAME: CHGCO.FOR
C      FUNCTION: READS AN ASCII PROGRAM FILE,
C      MODIFIES LINES 168-188
C      AND THEN REWRITES THE FILE BACK TO ORIGINAL FILE.
C*****
SUBROUTINE CHGCO
CHARACTER*72 PMLINE
CHARACTER*6 NEWCO
C
C***** OPEN FILES *****
C
C      '3DFILT.THS' IS THE MAIN FILTER PROGRAM WRITTEN IN
C      TMS320 ASSEMBLER. IT IS READ IN, MODIFIED, AND TRANSFERRED
C      TO 'NEW3D.ECK'.
C
OPEN(UNIT=1,FILE='3DFILT.THS',STATUS='OLD')
C
C***** *****
C
C      'NEWCO.TXT' CONTAINS THE 20 COEFFICIENTS AND SAMPLE RATE
C      VALUE TO BE PUT IN '3DFILT.THS'.
C
OPEN(UNIT=2,FILE='NEWCO.TXT',STATUS='OLD')
C
C***** *****
C
C      'NEW3D.ECK' IS THE NEW 3DFILTER PROGRAM IN TMS320
C      ASSEMBLER AFTER
C      ITS COEFFICIENTS AND SAMPLE RATE VALUES HAVE BEEN MODIFIED.
C      THE CARRIAGECONTROL='NONE' IS REQD TO MAKE FILE READABLE
C      BY TMS320 ASSEMBLER
C
OPEN(UNIT=3,FILE='NEW3D.ECK',CARRIAGECONTROL='NONE',
+     STATUS='NEW')
C*****
C
C      TRANSFER LINES 1-167 OF ORIGINAL FILE

```

```

DO 10 I=1,167
READ(1,25) PGMLINE
WRITE(3,25) PGMLINE
10    CONTINUE
C
C*****+
C
C      READ LINES 168-188 AND CHANGE COEFFS
C      THEN WRITE OUT TO NEW3D.ECK
C
C      DO 30 I=1,21
C      READ(1,25) PGMLINE
C      READ(2,40) NEWCO
C      PGMLINE(17:22)=NEWCO
C      WRITE(3,25) PGMLINE
30    CONTINUE
C
C*****+
C
C      TRANSFER REST OF ORIGINAL FILE
C
C      READ(1,25,END=50) PGMLINE
C      WRITE(3,25) PGMLINE
C      GO TO 35
35    CONTINUE
C
C*****+ FORMAT STATEMENTS ++++++
25    FORMAT(A)
27    FORMAT(1X,A)
40    FORMAT(1X,A6)
C*****+
C CLOSE ORIGINAL FILE AND DELETE
C
C      CLOSE(UNIT=1,DISP='DELETE')
C
C OPEN ORIGINAL FILE
C
C      OPEN(UNIT=1,FILE='3DFILT.THS',CARRIAGECONTROL=
C      + 'NONE',STATUS='NEW')
C
C OPEN TEMPORARY FILE THAT WILL PRINT OUT WHEN CLOSED
C      OPEN(UNIT=4,FILE='TEMP',STATUS='NEW')
C POINTER OF NEW3D.ECK TO BEGINNING OF FILE
C*****+
C      REWIND 3
C
C TRANSFER NEW3D.ECK BACK TO 3DFILT.THS
C
65    READ(3,25,END=70) PGMLINE
       WRITE(1,25) PGMLINE
C      WRITE(4,27) PGMLINE
C      GO TO 65
70    CONTINUE
C
C*****+

```

```

C ----- CLOSE FILES -----
C DELETE NEW3D.ECK
C
C     CLOSE(UNIT=3,DISP='DELETE')
C
C SAVE AND CLOSE 3DFILT.THS AND NEWCO.TXT
C
C     CLOSE (1)      !3DFILT.THS
C     CLOSE(UNIT=2,DISP='DELETE')
C
C PRINTABLE 3DFILT.THS AFTER MODIFICATION
C
C     CLOSE(UNIT=4,DISP='PRINT/DELETE')
C
C*****RETURN*****
C
C*****END*****
C
C*****SUB TO CALCULATE PROPER FREQ SPAN SETTING FOR B&K
C AND LENGTH OF TIME TO LEAVE 172B OUTPUT ON
C SUBROUTINE CALCFREQSPAN(ST,FREQS,ONTIME)
CHARACTER*5 FREQS
INTEGER ST
IF ((ST.GE..015).AND.(ST.LT..030)) THEN
    FREQS='25600' !32 MSECs
    ONTIME=.032
ELSEIF ((ST.GE..030).AND.(ST.LT..0625)) THEN
    FREQS='12800' !64 MSECs
    ONTIME=.064
ELSEIF ((ST.GE..0625).AND.(ST.LT..125)) THEN
    FREQS='6400' !.125 SECs
    ONTIME=.125
ELSEIF ((ST.GE..125).AND.(ST.LT..250)) THEN
    FREQS='3200' !.25 SECs
    ONTIME=.25
ELSEIF ((ST.GE..25).AND.(ST.LT..5)) THEN
    FREQS='1600' !.5 SECs
    ONTIME=.5
ELSEIF ((ST.GE..5).AND.(ST.LT.1.0)) THEN
    FREQS='800' !1 S
    ONTIME=1.0
ELSEIF ((ST .GE. 1.0).AND.(ST.LT.2.0)) THEN
    FREQS='400' !2 SECs
    ONTIME=2.0
ELSEIF ((ST .GE. 2.0) .AND. (ST .LT. 4.0)) THEN
    FREQS='200' !4 S
    ONTIME=4.0
ELSEIF ((ST .GE. 4.0) .AND. (ST .LT. 8.0)) THEN
    FREQS='100' !8 S
    ONTIME=8.0

```

```

ELSEIF ((ST .GE. 8.0) .AND. (ST .LT. 16.0)) THEN
    FREQB='50'      !16 S
    DNTIME=16.0
ELSEIF ((ST .GE. 16.0) .AND. (ST .LT. 32.0)) THEN
    FREQB='25'      !32 S
    DNTIME=32.0
ELSEIF ((ST .GE. 32.0) .AND. (ST .LT. 64.0)) THEN
    FREQB='12.5'    !64 S
    DNTIME=64.0
ELSEIF ((ST .GE. 64.0) .AND. (ST .LT. 128.0)) THEN
    FREQB='6.25'    !128 S
    DNTIME=128.0
ELSEIF ((ST .GE. 128.0) .AND. (ST .LT. 256.0)) THEN
    FREQB='3.125'   !256 S
    DNTIME=256.0
ELSEIF ((ST .GE. 256.0) .AND. (ST .LT. 512.0)) THEN
    FREQB='1.56'    !512 S
    DNTIME=512.0
ELSEIF (ST .GE. 512) THEN
    FREQB='1.56'
    DNTIME=512.0
ENDIF
RETURN
END

```

C*****
C END OF CALCFREQBSPAN
C*****

C

C*****
C CONVERTS A CHARACTER TO A NUMBER (0-9)
SUBROUTINE CHARTONUM(OPP, IOPT1)

```

CHARACTER*1 OPP
IOPT1=ICHAR(OPP)-48
RETURN
END

```

C*****
C SUBROUTINE TO CONVERT INPUT CHARACTER NUMBER TO NUMERIC
SUBROUTINE TWOCHAR(OPP2,AMP)

```

CHARACTER*2 OPP2
AMP=ICHAR(OPP2(1:1))-48.0
AMP=AMP+(ICHAR(OPP2(2:2))-48)/10.0
RETURN
END

```

C*****
SUBROUTINE THREECHAR(OPP3,STIME)

```

INTEGER STIME
CHARACTER*3 OPP3
STIME=0
IF (OPP3(1:1) .NE. ' ') THEN
    STIME=(ICHAR(OPP3(1:1))-48.0)*100
ENDIF
IF (OPP3(2:2) .NE. ' ') THEN
    STIME=STIME+(ICHAR(OPP3(2:2))-48)*10
ENDIF
STIME=STIME+(ICHAR(OPP3(3:3))-48)

```

RETURN
END

C*****
C CONVERTS SIX CHARACTERS TO INTEGER
C

SUBROUTINE SIXCHAR(OPP6,CLBW)

CHARACTER*6 OPP6

CLBW=0

CLBW=(ICHAR(OPP6(1:1))-48.0)*100000
CLBW=CLBW+(ICHAR(OPP6(2:2))-48.0)*10000
CLBW=CLBW+(ICHAR(OPP6(3:3))-48.0)*1000
CLBW=CLBW+(ICHAR(OPP6(4:4))-48.0)*100
CLBW=CLBW+(ICHAR(OPP6(5:5))-48.0)*10
CLBW=CLBW+(ICHAR(OPP6(6:6))-48.0)

RETURN

END

C*****
C PRINT PAIR.PRT FILE
C

SUBROUTINE PAIRPRT

OPEN(UNIT=8,FILE='PAIR.PRT',STATUS='OLD')
REWIND 8
CLOSE(UNIT=8,DISP='PRINT/DELETE')

RETURN

END

Appendix E.2 - TMS32010 Assembler List File

30FILT

300 FAMILY MACRO ASSEMBLER 2.1 63.076

11:43:22 10/23/85

0001 *

0002 *

0003 *

0004 *****

0005 *

0006 *

0007 *

LATEST VERSION: 26 June 1985

CASCADE STRUCTURE WITH SECOND ORDER 3-D SUBSECTIONS

0008 *

0009 *

0010 *

0011 *

0012 *

0013 *

0014 *

0015 *

0016 *

0017 *

0018 *

0019 *

0020 *

CAPT SCOTT ECKERT

$$\frac{Y(z)}{X(z)} = \frac{b0j + b1jz^{-1} + b2jz^{-2}}{1 - a1jz^{-1} - a2jz^{-2}}$$

0021 *

0022 *

0023 *

0024 *

0025 *

0026 *

0027 *

0028 *

0029 *

0030 *

0031 *

0032 *

0033 *

0034 *

0035 *

0036 *

0037 *

0038 *****

0039 *

0040 *

0041 *

0042 *

0043 *

0044 *

0045 *

0046 *

0047 *

0048 *

0049 *

0050 *

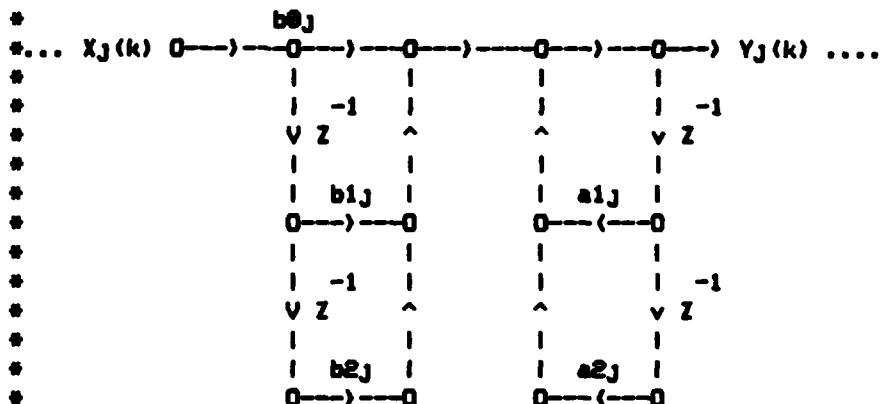
0051 *

0052 *

0053 *

0054 *

FILTER STRUCTURE



J= 1, 2, 3, 4

```

0055      *
0056      ****
0057      ****
0058      * Program Name: 3DFILT.ECK
0059      *
0060      * Functions: Provides the code to implement a general eighth-
0061      *          order cascade digital filter on a TMS32010
0062      *          microprocessor
0063      * Inputs: Input signal data from a 12-bit A/D converter
0064      *          with fullscale at +/- 10 volts.
0065      *
0066      * Outputs: To a 12-bit D/A converter that provides +/- 10 v
0067      *          full scale.
0068      *
0069      ****
0070      ****
0071      IDT '3DFILT'
0072      *
0073      *          INPUT DELAY NODE LOCATIONS IN DATA MEMORY
0074      *
0075      0000 X4K      EQU 0
0076      0001 X4KM1    EQU 1
0077      0002 X4KM2    EQU 2
0078      *
0079      0003 X3K      EQU 3
0080      0004 X3KM1    EQU 4
0081      0005 X3KM2    EQU 5
0082      *
0083      0006 X2K      EQU 6
0084      0007 X2KM1    EQU 7
0085      0008 X2KM2    EQU 8
0086      *
0087      0009 X1K      EQU 9
0088      000A X1KM1    EQU 10
0089      000B X1KM2    EQU 11
0090      *
0091      *          OUTPUT DELAY NODE LOCATIONS IN DATA MEMORY
0092      *
0093      000C Y4KM1    EQU 12
0094      000D Y4KM2    EQU 13
0095      000E Y3KM1    EQU 14
0096      000F Y3KM2    EQU 15
0097      0010 Y2KM1    EQU 16
0098      0011 Y2KM2    EQU 17
0099      0012 Y1KM1    EQU 18
0100      0013 Y1KM2    EQU 19
0101      *
0102      *          FILTER COEFFICIENT LOCATIONS IN DATA MEMORY
0103      ****
0104      *          FIRST SECTION
0105      0014 S01      EQU 20
0106      0015 S11      EQU 21

```

0107	0016	B21	EDU 22	
0108	0017	A11	EDU 23	
0109	0018	A21	EDU 24	
0110	*	SECOND SECTION		
0111	0019	B22	EDU 25	
0112	001A	B12	EDU 26	
0113	001B	B22	EDU 27	
0114	001C	A12	EDU 28	
0115	001D	A22	EDU 29	
0116	*	THIRD SECTION		
0117	001E	B23	EDU 30	
0118	001F	B13	EDU 31	
0119	0020	B23	EDU 32	
0120	0021	A13	EDU 33	
0121	0022	A23	EDU 34	
0122	*	FOURTH SECTION		
0123	0023	B24	EDU 35	
0124	0024	B14	EDU 36	
0125	0025	B24	EDU 37	
0126	0026	A14	EDU 38	
0127	0027	A24	EDU 39	
0128	*			
0129	*	OTHER VARIABLES		
0130	*			
0131	0028	CLOCK	EDU 40	'SAMPLE RATE
0132	0029	MODE	EDU 41	'SETS AIB MODE TO AUTO SAMPLE
0133	002A	MULT	EDU 42	'CONSTANT
0134	002B	MASK1	EDU 43	'INPUT DATA MASK
0135	002C	MASK2	EDU 44	'OUTPUT DATA MASK
0136	002D	YK	EDU 45	'FINAL FILTER OUTPUT
0137	002E	ONE	EDU 46	'VALUE ONE
0138	002F	MINUS1	EDU 47	'VALUE MINUS ONE (-1)
0139	0030	A	EDU 48	'TEMPORARY STORAGE LOCATION
0140	0031	B	EDU 49	'TEMPORARY STORAGE LOCATION
0141	*****			
0142	*			
0143	0000	ADRS 6		'START OF PROGRAM
0144	0000	F300	B START	
0001	001B			'BRANCH AROUND CONSTANTS
0145	*****			
0146	*	CONSTANT DATA		
0147	*****			
0148	*			
0149	*	COMPENSATOR COEFFICIENTS ARE	*	
0150	*	INITIALLY STORED IN PROGRAM MEMORY	*	
0151	*			
0152	*	THESE CONSTANTS ARE IN LINE IN ORDER TO BE ACCESSED BY		
0153	*	MAIN FORTRAN PROGRAM.		
0154	*	NOTE: IF LINES ARE ADDED INTO THIS PROGRAM BEFORE THIS		
0155	*	SECTION, THE MAIN PROGRAM DICES MUST BE MODIFIED. DICES		

0156 * COUNTS THE NUMBER OF LINES BEFORE THE CONSTANTS BELOW
 0157 * LINE LENGTH MUST ALSO BE LESS THAN 72 CHARS.
 0158 *
 0159 *
 0160 *
 0161 *
 0162 *
 0163 *
 0164 *
 0165 *
 0166 *
 0167 *(LINE 168 BELOW)
 0168 0002 32DF CB01 DATA 13023
 0169 0003 CD36 CB11 DATA -12970
 0170 0004 0000 CB21 DATA 0
 0171 0005 7F2E CA11 DATA 32702
 0172 0006 0000 CA21 DATA 0
 0173 0007 7FFF CB02 DATA 32767
 0174 0008 0000 CB12 DATA 0
 0175 0009 0000 CB22 DATA 0
 0176 000A 0000 CA12 DATA 0
 0177 000B 0000 CA22 DATA 0
 0178 000C 7FFF CB03 DATA 32767
 0179 000D 0000 CB13 DATA 0
 0180 000E 0000 CB23 DATA 0
 0181 000F 0000 CA13 DATA 0
 0182 0010 0000 CA23 DATA 0
 0183 0011 7FFF CB04 DATA 32767
 0184 0012 0000 CB14 DATA 0
 0185 0013 0000 CB24 DATA 0
 0186 0014 0000 CA14 DATA 0
 0187 0015 0000 CA24 DATA 0
 0188 0016 01A3 SMP DATA 419
 0189 *(LINE 168 ABOVE) *SAMPLE RATE (1/400TH OF
 0190 * DESIRED) THIS PROVIDES A RANGE
 0191 * OF 10 MSEC TO 20.97 SECs
 0192 0017 000A MD DATA >000A 'AUTO SAMPLE MODE
 0193 0018 0190 TH DATA 400 'CONSTANT
 0194 0019 7FFF M1 DATA >7FFF 'INPUT MASK VALUE
 0195 001A 0000 M2 DATA >0000 'OUTPUT MASK VALUE
 0196 *
 0197 #####END CONSTANT DATA#####
 0198 *
 0199 #####PROGRAM START#####
 0200 *
 0201 *
 0202 001B 6E00 START LDPK 0 'USE AR0 AS POINTER
 0203 *
 0204 *
 0205 * DATA STORAGE AND INITIALIZATIONS
 0206 *#####DATA STORAGE AND INITIALIZATIONS#####
 0207 001C 7E01 LACK 1 'ACC=1
 0208 001D 500E SACL ONE 'CONTENT OF DATA MEM "ONE" IS 1
 0209 001E 700C LARK AR0, MASK2 'THIS SECTION OF CODE LOADS
 0210 001F 7110 LARK AR1,24 'THE COMPENSATOR COEFFICIENTS

0211 0020 7E1A	LACK M2	'AND OTHER VALUES FROM PROGRAM		
0212 0021 6800	LOAD LARP AR0	'MEMORY TO DATA MEMORY		
0213 0022 6791	TBLR #-,AR1	'MV DATA FRM PRG MEM TO DATA MEM		
0214 0023 102E	SUB ONE	'DEC MEMORY POINTER BY ONE		
0215 0024 FE00	BNZ LOAD	'IF NOT AT END OF DATA, DO AGAIN		
0216	*			
0217	*	SET INITIAL STATE OF FILTER DELAY NODES TO ZERO		
0218	*			
0219 0026 7013	LARK AR0,Y1KM2	'LOAD ADDR OF Y1KM2		
0220 0027 7113	LARK AR1,19	'LOAD NUMBER OF LOCS TO ZERO OUT		
0221 0028 7F89	ZAC	'ACC=0		
0222 0029 6800	LP1	LP1	LARP AR0	'USE AR0 AS POINTER
0223 002A 5091	SACL #-,B,AR1	'STR ACC (-B) IN (AR0)/THEN USE AR1		
0224 002B F400	BANZ LP1	'DEC COUNTER (AR1) BY ONE		
0225 0029		'IF NOT AT END YET, KEEP GOING		
0226	*			
0227	*	SET UP MINUS1 TO -1		
0228	*			
0229 002D 2029	LAC MASK1			
0230 002E 612C	ADD8 MASK2			
0231 002F 502F	SACL MINUS1			
0232	*			
0233	*	INITIALIZE AIB		
0234	*			
0235 0030 4829	OUT MODE,PAB	'INITIALIZATION OF ANALOG		
0236 0031 4928	OUT CLOCK,PA1	'INTERFACE BOARD		
0237	*			
0238	*	SET OVERFLOW MODE		
0239	*			
0240 0032 7F88	SOVM	'SET AUTO OVERFLOW MODE		
0241	*			
0242	*****END DATA STORAGE AND INITIALIZATIONS*****			
0243	*			
0244	*****			
0245	*	START INFINITE LOOP		
0246	*****			
0247	*			
0248 0033 202A	WAIT LAC MULT	'READ INPUT EVERY 400 SAMPLES		
0249 0034 F600	WTMOR BIOZ NXTPT	'GO TO NXTPT WHEN BIO PIN GOES		
0035 0038		'LOW		
0250 0036 F900	B WTMOR	'GO WAIT FOR BIO PIN TO GO LOW		
0037 0034				
0251	*			
0252 0038 102E	NXTPT SUB ONE	'SUB ONE FROM CTR (ACC-ACC-1)		
0253 0039 4209	IN X1K,PAB	'READ SAMPLE		
0254 003A FE00	BNZ WTMOR			
0038 0034				
0255	*			
0256	*	CONVERT FORMAT TO TWO'S COMPLEMENT		
0257	*			
0258 003C 2009	LAC X1K	'THIS SECTION CONVERTS X1K FROM		
0259 003D 7029	XOR MASK1	'THE ANALOG INTERFACE BOARD'S		

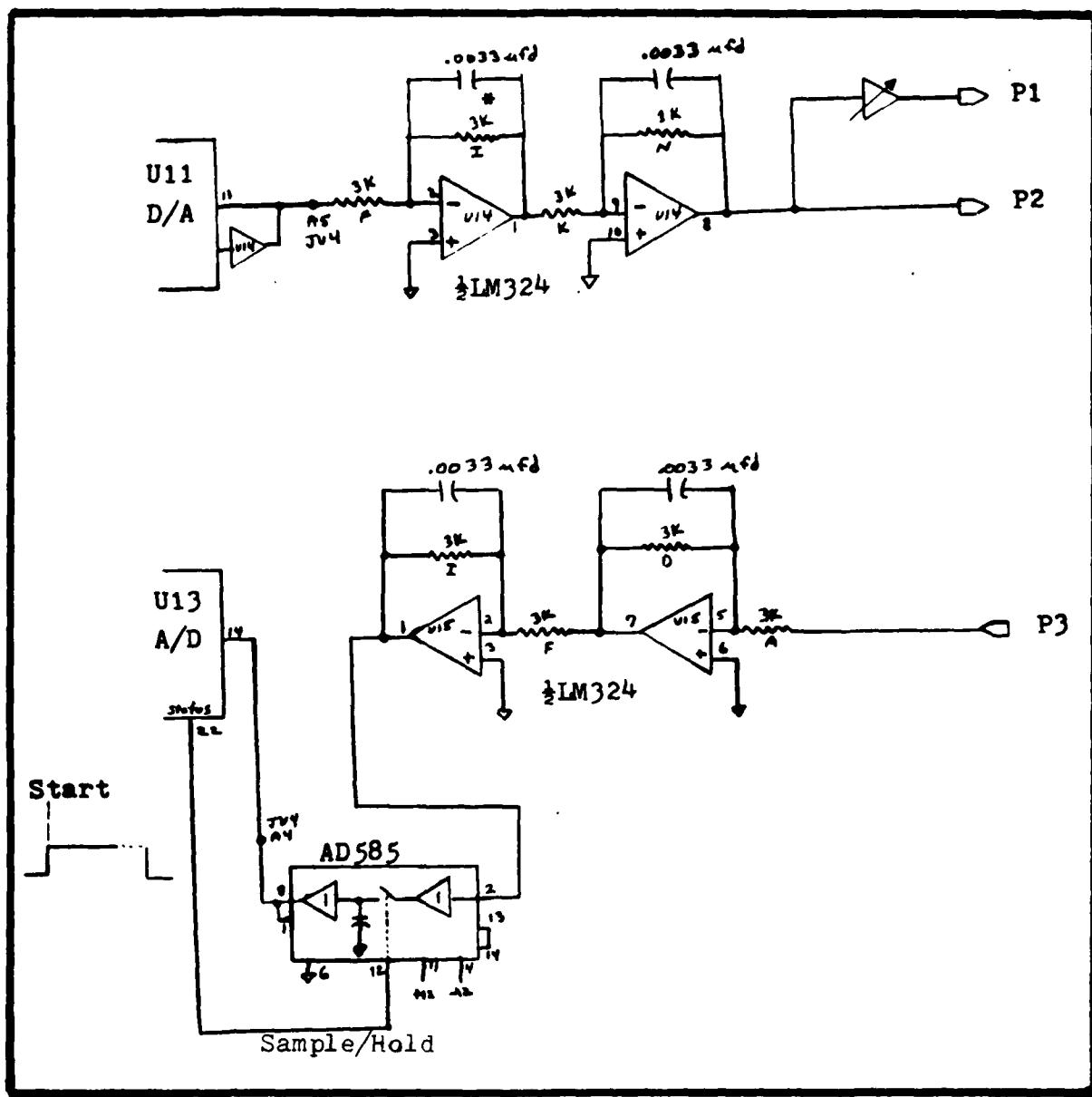
0260 003E 5809	SACL X1K	'FORMAT TO TWO'S COMPLEMENT
0261	*	'FORMAT
0262	*****	*****
0263	*	FIRST CASCADE SECTION
0264	*****	*****
0265	*	
0266 003F 7F89	ZAC	'ACC= 0
0267 0040 6A0B	LT X1KM2	'T=X1(K-2)
0268 0041 6D16	MPY B21	'P=T*B21/2
0269 0042 6B0A	LTD X1KM1	'T=X1(K-1); A=A+P; X1(K-2)=X1(K-1)
0270 0043 6D15	MPY B11	'P=T*B11/2
0271 0044 6B09	LTD X1K	'T=X1(K); A=A+P; X1(K-1)=X1(K)
0272 0045 6D14	MPY B01	'P=T*B01/2
0273 0046 6C13	LTA Y1KM2	'T=Y1(K-2)
0274 0047 6D18	MPY A21	'P=T*A21/2
0275 0048 6B12	LTD Y1KM1	'T=Y1(K-1); A=A+P; Y1(K-2)=Y1(K-1)
0276 0049 6D17	MPY A11	'P=T*A11/2
0277 004A 7F8F	APAC	'A=A+P
0278 004B 5912	SACH Y1KM1,1	'Y1(K-1)=Y1(K) (2*ACC)
0279	*	
0280	*****	*****
0281	*	SECOND CASCADE SECTION
0282	*****	*****
0283	*	
0284 004C 6A0B	LT X2KM2	'T=X2(K-2)
0285 004D 6D1B	MPY B22	'P=T*B22/2
0286 004E 6B07	LTD X2KM1	'T=X2(K-1); A=A+P; X2(K-2)=X2(K-1)
0287 004F 6D1A	MPY B12	'P=T*B12/2
0288 0050 6B06	LTD X2K	'T=X2(K); A=A+P; X2(K-1)=X2(K)
0289 0051 6D19	MPY B02	'P=T*B02/2
0290 0052 6C11	LTA Y2KM2	'T=Y2(K-2)
0291 0053 6D1D	MPY A22	'P=T*A22/2
0292 0054 6B1B	LTD Y2KM1	'T=Y2(K-1); A=A+P; Y2(K-2)=Y2(K-1)
0293 0055 6D1C	MPY A12	'P=T*A12/2

0294	0056	7F8F	APAC	' A=A+P
0295	0057	5910	SACH Y2KM1,1	' Y2(K-1)=Y2(K) (2=ACC)
0296	*			
0297	*****			*****
0298	*		THIRD CASCADE SECTION	
0299	*****		*****	
0300	*			
0301	0058	6A05	LT X3KM2	' T=X3(K-2)
0302	0059	6D08	MPY B23	' P=T+B23/2
0303	005A	6B04	LTD X3KM1	' T=X3(K-1); A=A+P; X3(K-2)=X3(K-1)
0304	005B	6D1F	MPY B13	' P=T+B13/2
0305	005C	6B03	LTD X3K	' T=X3(K); A=A+P; X3(K-1)=X3(K)
0306	005D	6D1E	MPY B03	' P=T+B03/2
0307	005E	6C0F	LTA Y3KM2	' T=Y3(K-2)
0308	005F	6D22	MPY A23	' P=T+A23/2
0309	0060	6B0E	LTD Y3KM1	' T=Y3(K-1); A=A+P; Y3(K-2)=Y3(K-1)
0310	0061	6D21	MPY A13	' P=T+A13/2
0311	0062	7F8F	APAC	' A=A+P
0312	0063	590E	SACH Y3KM1,1	' Y3(K-1)=Y3(K) (2=ACC)
0313	*			
0314	*****			*****
0315	*		FOURTH CASCADE SECTION	
0316	*****		*****	
0317	*			
0318	0064	6A02	LT X4KM2	' T=X4(K-2)
0319	0065	6D25	MPY B24	' P=T+B24/2
0320	0066	6B01	LTD X4KM1	' T=X4(K-1); A=A+P; X4(K-2)=X4(K-1)
0321	0067	6D24	MPY B14	' P=T+B14/2
0322	0068	6B00	LTD X4K	' T=X4(K); A=A+P; X4(K-1)=X4(K)
0323	0069	6D23	MPY B04	' P=T+B04/2
0324	006A	6C00	LTA Y4KM2	' T=Y4(K-2)
0325	006B	6D27	MPY A24	' P=T+A24/2

0326 006C 6B0C	LTD Y4KH1	'T=Y4(K-1);A=A+P;Y4(K-2)=Y4(K-1)
0327 006D 6D26	MDY A14	'P=T+A14/2
0328 006E 7F8F	APAC	'A=A+P
0329 006F 590C	SACH Y4KH1,1	'Y4(K-1)=Y4(K) (2*ACC)
0330	*	
0331	*****	
0332	***** END OF FOURTH CASCADE SECTION AND ENTIRE FILTER *****	
0333	*****	
0334 0070 592D	SACH YK,1	
0335 0071 292C	LAC MASK2	'THIS SECTION CONVERTS YK FROM
0336 0072 782D	XDR YK	'TWO'S COMPLEMENT FORMAT TO THE
0337 0073 392D	SACL YK	'ANALOG INTERFACE BOARD'S FORMAT
0338	*	
0339 0074 4A2D	OUT YK,PA2	'OUTPUT THE COMPENSATOR RESPONSE
0339 0074 4A2D	OUT YK,PA2	'OUTPUT THE COMPENSATOR RESPONSE
0340	*	
0341 0075 F900	B WAIT	'GO GET THE NEXT SAMPLE
0076 0033		
0342	*	
0343	END	

NO ERRORS, NO WARNINGS

Appendix F - Schematic Diagrams



* 3 Kohm resistors matched to within 1% per pair

Figure E-1. Analog Interface Board Modifications

Vita

Scott Burton Eckert was born on May 24, 1953 in Quantico, Virginia. He graduated from Bridgewater-Raritan High School - West, Bridgewater, New Jersey in 1971. Upon graduation, he entered the United States Air Force as a Precision Measuring Equipment Specialist assigned to Griffiss AFB, New York from 1972 - 1977. In 1977, he entered the Airmen's Education and Commissioning Program and graduated Magna cum Laude with a Bachelor of Science Degree in Computer Engineering from Syracuse University, Syracuse, New York in May, 1980. He entered Officer Training School at Lackland AFB, Texas in June of 1980 and graduated as a Distinguished Graduate August 15, 1980. Subsequent to graduation, he was assigned to Headquarters, Space Division, located at Los Angeles Air Force Station, California. He served as Chief, Systems Engineering Branch, Space Experiments Division, Space Defense Program Office for the Talon Gold Experiment, a DARPA-sponsored high precision space-based tracking and pointing experiment. Captain Eckert entered the School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, in May of 1984. He is a member of Eta Kappa Nu and Tau Beta Pi. Capt Eckert has a daughter, April.

Permanent Address: 796 Old Farm Road
Bridgewater, NJ
08807

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS							
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.							
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE									
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/85D-13		5. MONITORING ORGANIZATION REPORT NUMBER(S)							
6a. NAME OF PERFORMING ORGANIZATION School of Engineering	6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION							
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433		7b. ADDRESS (City, State and ZIP Code)							
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER							
8c. ADDRESS (City, State and ZIP Code)		10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. PROJECT NO. TASK NO. WORK UNIT NO.							
11. TITLE (Include Security Classification) See Box 19									
12. PERSONAL AUTHOR(S) Scott B. Eckert, B.S., Capt, USAF									
13a. TYPE OF REPORT MS Thesis	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1985 December	15. PAGE COUNT 417						
16. SUPPLEMENTARY NOTATION									
17. COSATI CODES <table border="1"><tr><th>FIELD</th><th>GROUP</th><th>SUB. GR.</th></tr><tr><td>09</td><td>02</td><td></td></tr></table>		FIELD	GROUP	SUB. GR.	09	02		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Digital Controllers, Digital Compensators, Digital Filters, Controller Implementation, Hybrid Simulation, IEEE-488 Bus.	
FIELD	GROUP	SUB. GR.							
09	02								
19. ABSTRACT (Continue on reverse if necessary and identify by block number) Title: Development of a Digital Interactive Controller Evaluation System (DICES) Thesis Chairman: Dr. Gary B. Lamont Professor of Electrical Engineering									
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED							
22a. NAME OF RESPONSIBLE INDIVIDUAL Gary B. Lamont		22b. TELEPHONE NUMBER (Include Area Code) 513-255-5533	22c. OFFICE SYMBOL AFIT/ENG						

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Block 19 (Cont)

The Digital Interactive Controller Evaluation System (DICES) is a user interactive system which permits the implementation of digital controller designs based on the TMS32010 digital-signal-processing microprocessor for a given single input-single output plant model. DICES partitions the controller design into second-order 3D filter sections and quantizes the coefficients. These coefficients are loaded into a generic filter program written in TMS32010 assembly language, which are then assembled and loaded into the TMS32010 for execution. The controller can be placed in the forward or feedback path of an analog computer system which reflects the plant model. Performance data is obtained via IEEE-488 controlled instruments under control of a VAX 11/780.

Request, in draft

SECURITY CLASSIFICATION OF THIS PAGE

END

FILMED

386

DTIC